MASTER EN SCIENCES MATHÉMATIQUES

Trois approches de classification automatique non supervisée basées sur un Processus de Poisson non homogène dans le plan

Jeannee, N.

Award date:
1996

Facultés Universitaires Notre-Dame De La Paix
Namur
Faculté des Sciences

Trois approches de classification automatique non supervisée basées sur un processus de Poisson non homogène dans le plan

Mémoire présenté pour l'obtention du grade
de Licencié en Sciences
mathématiques
par

Promoteur: J.-P. RASSON

Nicolas JEANNEE

Année académique: 1995-1996

Je tiens tout d'abord à remercier mon promoteur Monsieur Jean-Paul Rasson, pour sa patience, son dynamisme et son implication dans ce mémoire.
Merci à Monsieur Vincent Bertholet pour sa perpétuelle disponibilité et ses judicieux conseils.
Merci également à Monsieur Tite Kubushishi pour l'intérêt qu'il a manifesté dans mes "recherches", ses remarques constructives et ses encouragements.
Merci à Mesdemoiselles Fabienne Montaigne et Sandrine Lissoir pour leur bonne humeur et leur serviabilité.
Merci aussi à Françoise Gérard, Christel Requier et Fabrice Rasir, mes compagnons de fortune, pour leur soutien.
Merci finalement à tous ceux que je n'ai pas nommé mais qui savent tout ce que je leur dois.

Trois approches de classification automatique non supervisée basées sur un processus de Poisson non homogène dans le plan.

Résumé

Ce mémoire consiste en l'évaluation de trois approches pour la classification automatique de points résultant d'un processus de Poisson non homogène sur une union de domaines disjoints du plan euclidien.

La première approche est basée sur une recherche de la classification optimale, par technique de convexité, à partir d'une classification initiale, et ce par échange de points entre classes.
Ensuite, une approche fondée sur l'utilisation de la connexité comme alternative à la convexité, en recourrant à la théorie des graphes, est envisagée.
Finalement, une approche géométrique toujours basée sur la connexité, reposant sur l'accroissement de boules centrées aux points afin de les regrouper lorsque ces boules s'intersectent est présentée.

En outre, une discussion du choix du paramètre de lissage en estimation d'intensité, en utilisant l'analyse discriminante, est faite.

Abstract

The goal of this work is to evaluate three approaches for the clustering of points supposed to be distributed according to a non-homogeneous Poisson process on the union of disjoints subdomains of the euclidean plane.

The first approach is based on the search of the optimal classification, by a convex technique, starting from an initial one, by switching points between clusters.
Then, a second approach based on the use of connexity, using graph theory, is explained.
A last geometric approach, still resting on connexity, based on the growth of spheres centered at points in order to regroup them when the spheres intersect themselves, is given.

Besides, the choice of the smoothing parameter in intensity estimation, using discriminant analysis, is discussed.

Table des matières

Introduction

Les recherches dans le cadre desquelles ce mémoire s'inscrit ont commencé en 1977 avec la question posée par D.G. Kendall de l'estimation d'un convexe à partir d'observations intérieures résultant d'un processus de Poisson homogène. La solution, trouvée par Ripley B.D. et Rasson J.-P. [15], repose sur une dilatation de l'enveloppe convexe des observations à partir de leur centroïde. Ce résultat a ensuite été appliqué en classification automatique non supervisée où, si les observations résultent d'un processus de Poisson homogène sur l'union de m domaines convexes disjoints, la solution du maximum de vraisemblance donnée par Hardy A. et Rasson J.-P. [10] consiste à "trouver la partition des observartions en m classes qui minimise la somme des mesures de Lebesgue des enveloppes convexes de ces classes". Le critère correspondant en analyse discriminante, donné par Baufays P. et Rasson J.-P. [2, 3, 4], affecte le nouveau point à la classe pour laquelle il minimise la mesure de Lebesgue ajoutée à l'enveloppe convexe de la classe.

Ce critère ne permettant pas dans les applications de classer les points appartenant à plus d'une enveloppe convexe, on a eu recours à un processus ponctuel plus général, le processus de Poisson non homogène. Dans les solutions données ci-dessus, cela a conduit à remplacer la mesure de Lebesgue par la notion d'intensité intégrée, et a permis la classification des points appartenant à plus d'une enveloppe convexe.

Dans le contexte de la classification automatique non supervisée, une question importante consiste alors à savoir, sous cette hypothèse de processus de Poisson non homogène, comment atteindre la solution donnée. C'est l'objectif de ce mémoire, qui envisage dans ce but trois approches.

Le problème de la détermination du nombre de classes d'une classification, complexe et extrêmement important, n'est pas abordé ici. Nous considérerons ce nombre fixé pour chaque exemple.

Avant d'aborder ces approches, nous présentons dans les deux premiers chapitres les notions de base nécessaires à la compréhension de la suite du mémoire. Dans un premier

temps, les principes de la Classification sont brièvement rappelés. Le second chapitre détaille notre hypothèse de départ relative aux observations, i.e. le processus de Poisson non homogène.

Ce dernier entraînant la nécessité d'une estimation de l'intensité préalable à toute stratégie de classification automatique non supervisée, le troisième chapitre décrit l'estimateur d'intensité que nous utiliserons.

La première approche présentée repose sur l'optimisation d'une classification initiale par échange de points entre classes, en vue d'obtenir une classification optimale. Cette approche utilise des techniques découlant de l'hypothèse de convexité des classes qui est faite.

Les deux approches suivantes sont basées sur l'utilisation de la connexité pour le regroupement des points, la convexité semblant une hypothèse trop restrictive. Une approche exploite les outils de la théorie des graphes. La seconde repose sur des considérations plus géométriques. Elle se base sur la croissance de boules centrées aux observations, "connectant" des observations lorsque les boules correspondantes s'intersectent.

Finalement, nous étudions quel intérêt il peut y avoir à utiliser l'analyse discriminante pour l'estimation du paramètre de lissage nécessitée par l'estimation de l'intensité du processus de Poisson non homogène.

Chapitre 1

Classification

Nous commençons par un chapitre donnant une vue d'ensemble de la Classification, de ses deux branches principales que sont le *clustering*[1] et l'*Analyse discriminante*. Cette dernière n'intervenant que très ponctuellement par la suite, nous nous contenterons d'en rappeler le principe.

1.1 Clustering

1.1.1 Position du problème

Considérons n objets $k = 1, ..., n$ $(n \geq 1)$ et $\mathcal{O} = \{1, ..., n\}$ l'ensemble de ces objets. A partir de données connues caractérisant les propriétés de ces objets (voir 1.1.3), nous cherchons à regrouper les n objets dans des sous-ensembles $C_1, ..., C_m$ $(m \geq 1)$ de $\mathcal{O}$ de sorte que chacun des C_i regroupe des objets similaires, et que les différents sous-ensembles soient bien séparés.

Définitions :

- Un tel sous-ensemble C_i $(1 \leq i \leq m)$ de $\mathcal{O}$ est appelé une *classe* (ou *cluster*).
- La famille $\mathcal{C} = (C_1, ..., C_m)$ obtenue à partir des classes $C_1, ..., C_m$ est appelée *classification* de $\mathcal{O}$.

[1] *Clustering* est la traduction anglo-saxonne de *classification automatique non supervisée*. Nous emploierons indifféremment les deux termes dans ce mémoire.

1.1.2 Types de classifications

Bien que cela ne soit pas exhaustif, nous pouvons distinguer essentiellement quatre types de classifications.

1. Partitions
   Une *partition* $\mathcal{C} = (C_1, \ldots, C_m)$ $(m \geq 1)$ de $\mathcal{O}$ est une famille de classes disjointes, non-vides recouvrant $\mathcal{O}$,

$$\begin{cases} C_i \cap C_j \neq \emptyset \quad \text{pour } i \neq j \ , \\ \bigcup_{i=1}^{m} C_i = \mathcal{O} \ , \\ C_i \neq \emptyset \ , \qquad 1 \leq i \leq m. \end{cases}$$

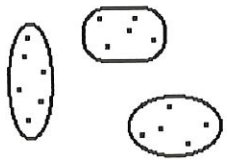


Figure 1.1: Partition d'un ensemble de données en 3 classes.

L'objectif est ici d'obtenir une *partition optimale* $\mathcal{C}^\star$ de $\mathcal{O}$, c'est-à-dire résultant de l'optimisation d'un *critère de classification* $g(\mathcal{C})$.
Certains critères nécessitent un nombre m de classes fixé, mais ce n'est pas le cas pour tous.

Soit $\mathcal{P}$ l'ensemble des partitions de $\mathcal{O}$. Si nous définissons $\mathcal{P}_m$ comme l'ensemble des partitions en m classes de $\mathcal{O}$, nous avons

$$\mathcal{P} = \bigcup_{m \geq 1} \mathcal{P}_m.$$

Exemple de critère : En définissant le *diamètre d'une classe* $C \subseteq \mathcal{O}$ comme

$$d(C) = \max_{k,l \in C} \{d_{kl}\},$$

nous obtenons par exemple le *critère du diamètre minimum* :

$$\min_{\mathcal{C} \in \mathcal{P}_m} g(\mathcal{C}) \ ,$$

où $g(\mathcal{C}) = \max_{i=1,\ldots,m} d(C_i)$, et $m \geq 2$ est fixé.

2. Hiérarchies

Définition : $\mathcal{C} = (A, B, C, \ldots) \stackrel{not}{=} \mathcal{H}$ est une *hiérarchie* de $\mathcal{O}$ ssi

i. $\mathcal{O} \in \mathcal{H}$,

ii. $\forall j \in \mathcal{O},\ \{j\} \in \mathcal{H}$,

iii. $\forall A, B \in \mathcal{H},\ A \cap B \in \{\emptyset, A, B\}$.

Nous allons ici construire une hiérarchie de partitions.
Soit une fonction monotone $h : \mathcal{H} \longrightarrow \mathbb{R}_+$ telle que

. $\forall A, B \in \mathcal{H},\ A \subseteq B \Rightarrow h(A) \leq h(B)$,

. $\forall k \in \mathcal{O},\ h(k) = 0$.

Le couple $(\mathcal{H}, h)$ est appelé *dendogramme*, ou *hiérarchie indicée.*

Pour tout niveau $h \geq 0$, une *classification de niveau* h est définie par

$$C(h) = \{A \in \mathcal{H} \text{ tq } f(A) \leq h \text{ et } \nexists B \in \mathcal{H} \text{ tq } A \subset B \Rightarrow f(B) \leq h\}.$$

C'est l'ensemble des classes formées au niveau h.

Dans l'exemple de la figure 1.2, nous avons la classification de niveau h_0 suivante :

$$C(h_0) = \{\underbrace{\{1,2\}}_{C_1}, \underbrace{\{3\}}_{C_2}, \underbrace{\{4,5,6\}}_{C_3}\}.$$

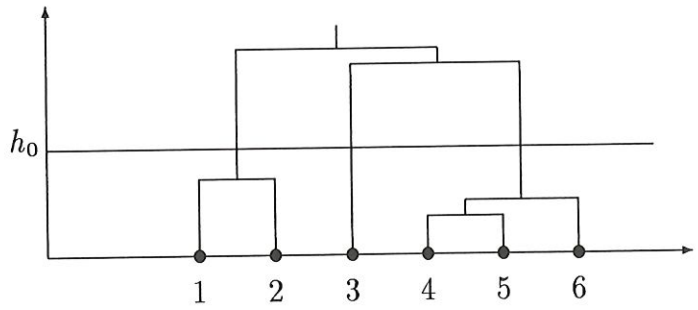


Figure 1.2: Classification de niveau h_0.

Il y a deux démarches possibles pour obtenir une hiérarchie de partitions. La première part de n classes, chaque classe étant un *singleton-objet* de $\mathcal{O}$, et regroupe à chaque étape les deux classes les plus similaires, afin d'obtenir par exemple le nombre de classes désiré (*Algorithmes agglomératifs*).
La seconde stratégie, au contraire, part d'une classe unique $\mathcal{O}$, et divise à chaque étape une classe en deux de manière à maximiser l'hétérogénéité inter-classes (*Algorithmes divisifs*).

3. Recouvrements

   Les classes de $\mathcal{C} = (C_1, \ldots, C_m)$ doivent ici être non-vides, mais il n'est plus n'exigé qu'elles soient disjointes.

   Si $\bigcup_{i=1}^{m} C_i = \mathcal{O}$, i.e. tout élément est classé, on parlera de *classification exhaustive.*

   Par opposition, dans le cas contraire où $\bigcup_{i=1}^{m} C_i \subset \mathcal{O}$, on parlera de *classification non-exhaustive.*

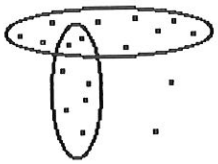


Figure 1.3: Exemple de classification non-exhaustive.

4. Classifications floues

   Supposons que nous ayons n objets à regrouper en m classes. Plutôt que d'affecter chaque élément à une classe, nous allons ici chercher à obtenir un pourcentage pour chaque objet d'être attribué à chaque classe.

   Nous aurons donc une matrice $U = (u_{ik})_{m \times n}$, où $u_{ik} \equiv$ pourcentage avec lequel l'objet k est attribué à la classe i.

1.1.3 Types de données

1. Vecteurs d'observation

   On mesure pour chaque objet p variables $X^1, ..., X^p$ $(p \geq 1)$,
   où $X^j \in \mathcal{X}^j$, $j = 1, ..., p$.
   Un objet k se caractérisé donc par

$$X_k = \begin{pmatrix} X_{k1} \\ \vdots \\ X_{kp} \end{pmatrix} \in \mathcal{X}^1 \times \mathcal{X}^2 \times ... \times \mathcal{X}^p.$$

   Cela nous donne la $(n \times p)$-matrice de données

$$X = \begin{pmatrix} X_{11} & \ldots & X_{1p} \\ \vdots & \ddots & \vdots \\ X_{n1} & \ldots & X_{np} \end{pmatrix} = \begin{pmatrix} X_1^\top \\ \vdots \\ X_n^\top \end{pmatrix}.$$

Remarques: On distingue quatre types classiques de variables:

. quantitatives (ex: $\mathcal{X}_j = \mathbb{R}$),

. binaires (ex: $\mathcal{X}_j = \{0,1\}$, $\mathcal{X}_j = \{\text{oui}, \text{non}\}$),

. nominales (ex: $\mathcal{X}_j = \{0,1,\ldots,s_j\}(s_j \geq 1)$),

. ordinales (ex: $\mathcal{X}_j = \{a,b,\ldots,f\}$ où $a < \ldots < f$, et $X^j \equiv$résultat d'un examen).

Différents types de variables peuvent être mélangés dans une même matrice de données. Ces *données mixtes* compliquent considérablement la théorie de la classification.

2. Dissimilarités

Définition: Une *dissimilarité* d sur $\mathcal{O}$ est une fonction

$$\begin{array}{rccl} d & : & \mathcal{O} \times \mathcal{O} & \longrightarrow & \mathbb{R}_+ \\ & & (k,l) & \rightsquigarrow & d(k,l) \overset{not}{=} d_{kl}, \end{array}$$

telle que:

. $d_{kk} = 0$, $\forall k \in \mathcal{O}$,

. $0 \leq d_{kl} = d_{lk} < +\infty$, $\forall k,l \in \mathcal{O}$.

Pour tout couple (k,l) d'objets de $\mathcal{O}$, d_{kl} est appelé *indice de dissimilarité* entre k et l.

Exemple: Pour des données quantitatives, nous avons la *norme infinie* (ou *distance du maximum*)

$$\forall k,l \in \mathcal{O},\ d_{kl} = \|X_k - X_l\|_\infty \overset{\text{def}}{=} \max_{j=1,\ldots,p} |X_{kj} - X_{kl}|.$$

3. Similarités

Définition: Une *similarité* s sur $\mathcal{O}$ est une fonction

$$\begin{array}{rccl} s & : & \mathcal{O} \times \mathcal{O} & \longrightarrow & [0,1] \\ & & (k,l) & \rightsquigarrow & s(k,l) \overset{not}{=} s_{kl}, \end{array}$$

telle que: $0 \leq s_{kl} = s_{lk} \leq s_{kk} = 1$, $\forall k,l \in \mathcal{O}$.

De la même manière que pour une dissimilarité, s_{kl} représente l'*indice de similarité* entre les objets k et l de $\mathcal{O}$.

Il est possible d'obtenir une similarité à partir d'une dissimilarité (ex: $s = e^{-d}$), et inversément (ex: $d = 1 - s$).

1.1.4 Objets utilisés

Les objets considérés dans ce mémoire sont des réalisations d'un processus de Poisson non homogène sur une union de domaines convexes disjoints du plan euclidien. Nous reviendrons en détail sur ce processus au chapitre suivant. Le problème consiste alors à retrouver ces domaines.
Nous noterons à présent $\mathcal{O} = \{x_1, \ldots, x_n\}$.

1.2 Analyse discriminante

Contrairement à la classificaton automatique non supervisée où nous ne connaissons rien quant à l'appartenance des objets à des classes, nous disposons ici d'une *base d'entraînement.* Autrement dit, nous connaissons la classe d'appartenance de chacun des n objets de la base, en plus des propriétés de ceux-ci.

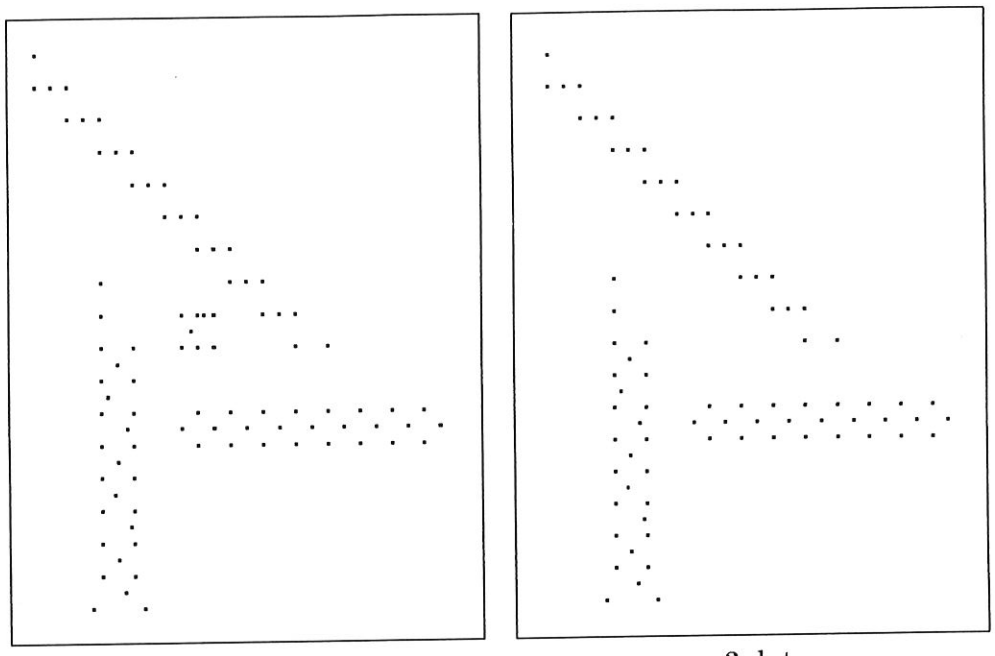
Face à un nouvel objet dont nous connaissons les propriétés, le problème consiste alors à déterminer la classe à laquelle nous affectons l'objet.

Il faut donc, à partir des échantillons d'entraînement, dégager une *règle d'affectation.* Nous aurons l'occasion d'en voir deux exemples au chapitre suivant.

1.3 Exemples

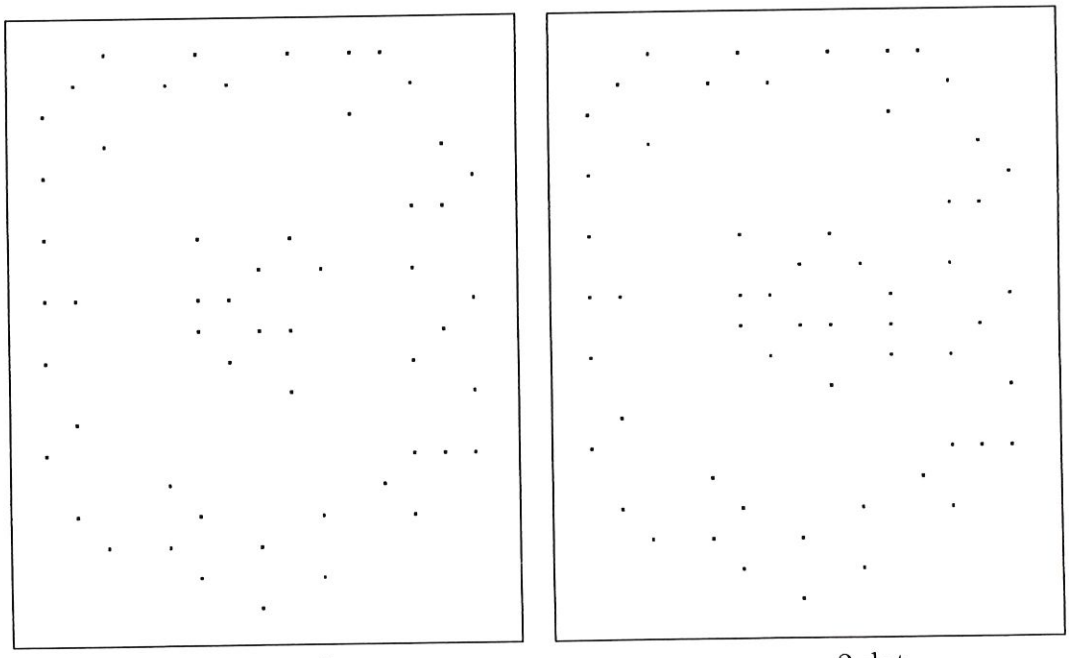
Nous présentons dans cette section les exemples qui serviront à l'expérimentation des méthodes présentées dans ce mémoire.

L'exemple r4.dat est classique. Sa particularité vient du fait qu'il présente des classes non linéairement séparables. Remarquons que deux points de la classe centrale sont confondus. L'exemple r3.dat est obtenu à partir de r4.dat par suppression de cette classe centrale.



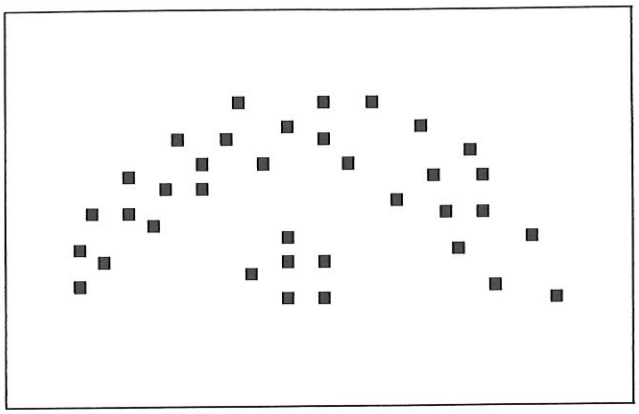
r4.dat r3.dat

couronne2.dat est identique à couronne.dat, excepté les trois points situés sur le bord droit de la classe centrale.

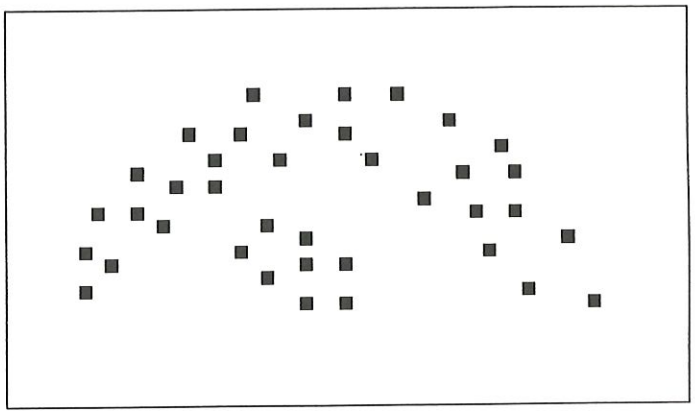


couronne.dat couronne2.dat

noncv_36.dat présente une classe non convexe. Cet exemple est identique à noncv_38.dat, ce dernier ayant deux points de plus à la périphérie de la classe centrale.

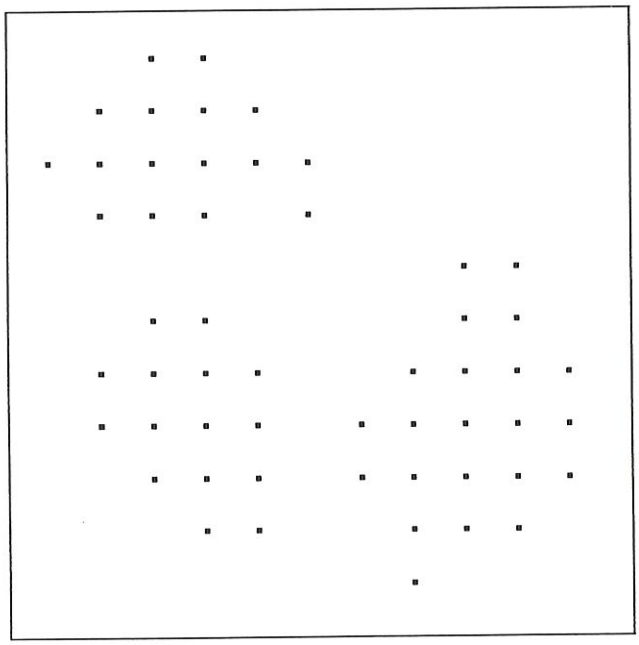


noncv_36.dat

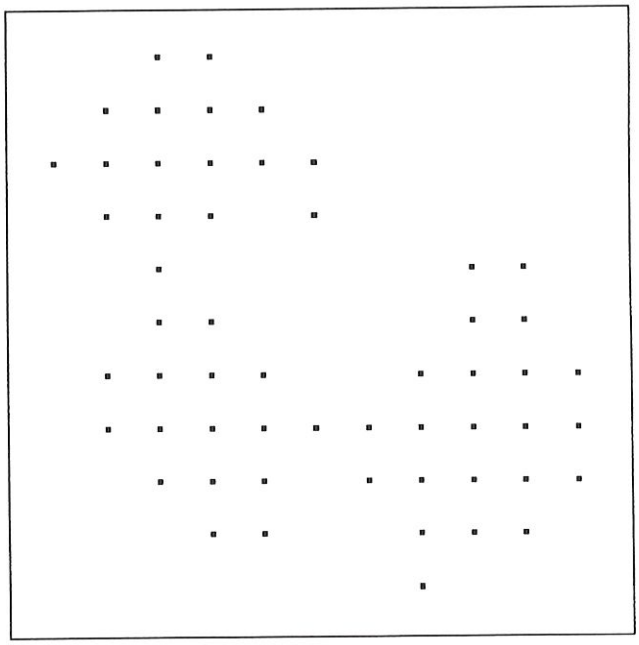


noncv_38.dat

r5.dat présente la particularité d'être fort symétrique. Il est obtenu à partir de r1.dat, exemple également classique, auquel on a retiré les deux points frontières entre les classes.



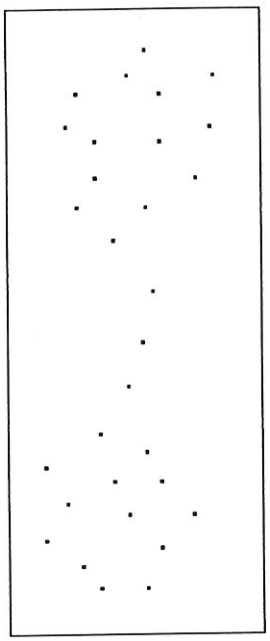
r5.dat



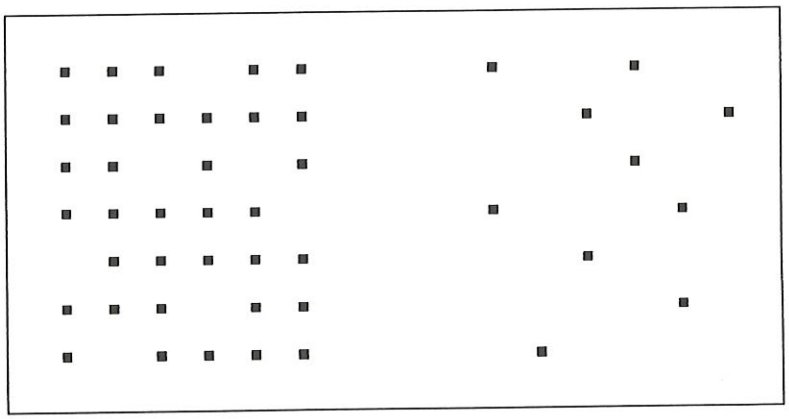
r1.dat

cluster2.sas présente deux classes chaînées.

L'exemple clust_45.ex5 nous semble intéressant car, bien qu'apparemment simple, il présente deux classes fort différentes : l'une très dense, l'autre beaucoup moins.

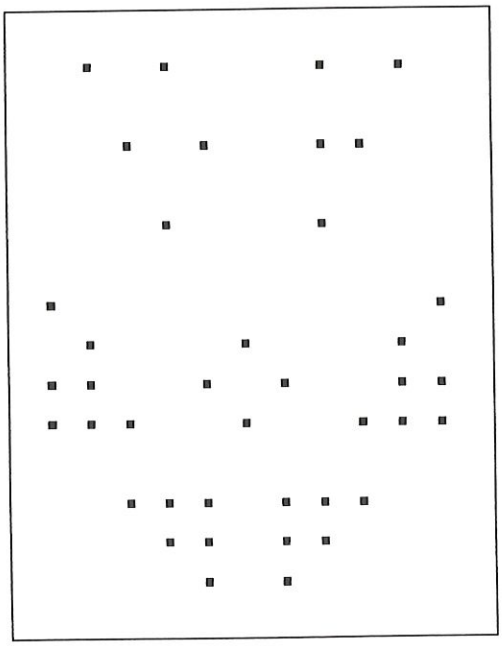


cluster2.sas



clust_45.ex5

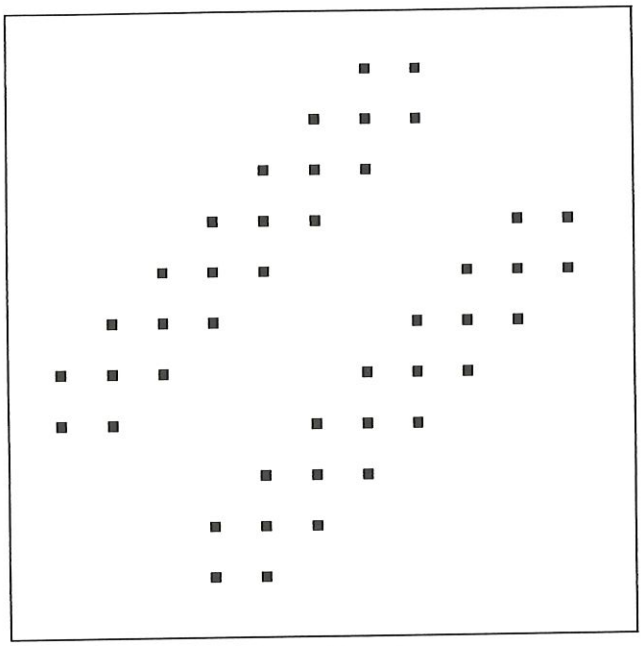
L'exemple clust_40.ex3 comporte sept classes.



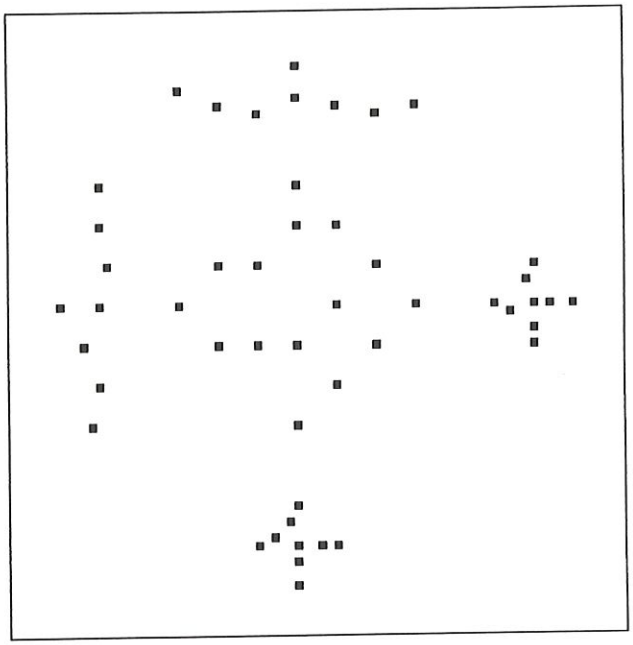
clust_40.ex3

clust_44.ex4 est un exemple de classes allongées.

Finalement, clust_49.ex13 a apparemment quatre classes périphériques, qui entourent une classe centrale peu dense.



clust_44.ex4



clust_49.ex13

Notons que dans les classifications présentées dans ce mémoire, si certains points ne sont pas entourés explicitement, afin de ne pas alourdir la représentation, ils forment quand même une classe entre eux.

Chapitre 2

Processus de Poisson

Les objets que nous cherchons à classer sont des points du plan euclidien qui résultent d'un processus de Poisson non homogène sur des domaines convexes disjoints du plan euclidien.

Ce chapitre rappelle ce qu'il faut entendre par processus de Poisson homogène et non homogène. Parce qu'ils nous seront utiles par la suite, nous donnerons pour chacun de ces processus un critère de classification et une règle d'affectation.

2.1 Processus de Poisson homogène (ou stationnaire)

2.1.1 Définition

Deux propriétés définissent un processus de Poisson homogène.

1. Les variables aléatoires comptant le nombre de points dans des domaines disjoints du plan euclidien sont stochastiquement indépendantes.
2. Le nombre de points d'un domaine borné D suit une distribution de Poisson de paramètre $\lambda m(D)$, $m(D)$ désignant la mesure de Lebesgue de D[1]. λ est appelé *intensité du processus.*

[1]L'espérance d'une loi de Poisson étant son paramètre, cela signifie que le nombre moyen de points d'un domaine est proportionnel à la mesure de Lebesgue de ce domaine.

2.1.2 Critère de classification

Nous supposons donc d'abord que les points de $\mathcal{O}$ sont générés par un processus de Poisson homogène sur $D = \bigcup_{k=1}^{m} D_k$, les domaines D_k étant supposés disjoints.

Il en résulte que ces points sont indépendamment et uniformément distribués sur D. La fonction de vraisemblance du vecteur d'observation prend donc la forme

$$F_D((x_1, \ldots, x_n)) = \prod_{i=1}^{n} \frac{1_D(x_i)}{m(D)} = \frac{1}{(m(D))^n} \prod_{i=1}^{n} 1_D(x_i),$$

où $m(D) = \sum_{k=1}^{m} m(D_k)$ est la mesure de Lebesgue de D. Cette vraisemblance sera maximale pour le domaine D dont la mesure de Lebesgue sera minimale.
La convexité des D_k est imposée pour que D soit acceptable, afin d'éviter des solutions triviales.

La solution du maximum de vraisemblance consiste à

> *trouver la partition en m groupes qui minimise la somme des mesures de Lebesgue des enveloppes convexes de ces groupes.*

2.1.3 Règle d'affectation associée en analyse discriminante

La distribution conditionnelle de la k-ième population est supposée uniforme sur le domaine convexe D_k, et la probabilité a priori p_k qu'un point appartienne à la k-ième population est proportionnelle à la mesure de Lebesgue de D_k,

$$p_k = \frac{m(D_k)}{m(D)}.$$

Nous avons les densités

$$f_k(x) = \frac{1_{D_k}(x)}{m(D_k)},$$

et

$$f(x) = \sum_{k=1}^{m} p_k f_k(x) = \frac{\sum_{k=1}^{m} 1_{D_k}(x)}{m(D)}.$$

Comme règle de décision, nous considérons la règle bayésienne, où les domaines D_k sont remplacés par leurs estimations du maximum de vraisemblance.

Soit X_k l'ensemble des points de la population k, $H(X_k)$ l'enveloppe convexe de X_k, et x un point à affecter à l'une des m populations.
Si x est affecté au k-ième groupe, l'estimation des D_j est

$$\hat{D}_j = \begin{cases} H(X_j) & \text{si } j \neq k\,, \\ H(X_j \cap \{x\}) & \text{si } j = k\,. \end{cases}$$

L'estimation du maximum de vraisemblance des $p_k f_k$ est donc pour k donné

$$\hat{p_k f_k}(x) = \frac{1_{\hat{D}_k}(x)}{\sum_{j=1}^{m} m(H(X_j)) + S_k(x)},$$

où $S_k(x) = m(H(X_k \cap \{x\})) - m(H(X_k))$ représente la mesure de Lebesgue ajoutée par x à l'enveloppe convexe de X_k.

Pour que les domaines restent disjoints, il faut, pour affecter x à une classe k, que

$$H(X_k \cup \{x\}) \cap H(X_j) = \emptyset\,, \text{ pour } j \neq k\,.$$

Si ce n'est pas le cas, on pose $S_k(x) = +\infty$.
Autrement dit, les enveloppes convexes doivent rester disjointes.

La règle d'affectation est donc

x est affecté à la k-ième population ssi $S_k(x) < S_j(x)$, $\forall j \neq k$,

c'est-à-dire

x est affecté à la population k pour laquelle x minimise la mesure de Lebesgue qu'il ajoute à l'enveloppe convexe de cette population.

2.2 Processus de Poisson non homogène

2.2.1 Définition

Ce processus est une généralisation du précédent en ce sens que le paramètre λ est remplacé ici par une fonction intensité $q(x)$. Les propriétés du processus deviennent

1. Ici encore, les V.A. de comptage sur des domaines disjoints sont stochastiquement indépendantes.
2. Le nombre de points d'un domaine borné D suit une distribution de Poisson de paramètre $m(D)$, où

$$m(D) = \int_D q(x)\,dx.$$

Donc, suivant le processus, $m(D)$ désignera la mesure de Lebesgue du domaine D ou l'intensité intégrée sur ce domaine.

2.2.2 Critère de classification

Les points sont donc générés par un processus de Poisson non homogène d'intensité q sur l'union D de m domaines D_k disjoints. Comme dans le cas homogène, nous cherchons la solution du maximum de vraisemblance pour D. La vraisemblance du vecteur d'observation s'écrit ici

$$F_D((x_1,\ldots,x_n)) = \frac{1}{(m(D))^n}\prod_{i=1}^{n} 1_D(x_i)q(x_i).$$

Si l'intensité est connue ou estimée, la solution est, comme l'estimation du maximum de vraisemblance d'un domaine convexe borné est encore l'enveloppe convexe,

trouver la partition en m groupes qui minimise la somme des intensités intégrées des enveloppes convexes de ces groupes.

Autrement dit, nous cherchons la partition $\mathcal{C}^\star = (C_1^\star, ..., C_m^\star)$ pour laquelle

$$g(\mathcal{C}^\star) = \min_{\mathcal{C} \in \mathcal{P}_m} g(\mathcal{C}) \ ,$$

où $g(\mathcal{C}) = \sum_{i=1}^{m} \int_{C_i} q(x)dx$, si $q(x)$ est l'intensité du processus de Poisson non homogène.

2.2.3 Règle d'affectation associée en Analyse Discriminante

Il est possible de reconstruire la règle de discrimination associée à ce processus en procédant comme dans le cas homogène. Contentons-nous de l'énoncer dans le cas où les intensités q_k ne dépendent pas de k, en considérant un point x à affecter à l'une des k populations :

affecter x à la population k pour laquelle x minimise l'intensité intégrée ajoutée à l'enveloppe convexe de cette population.

Chapitre 3

Estimation de l'intensité

Nous avons supposé au premier chapitre que les données que nous observons résultent d'un processus de Poisson non homogène, d'intensité notée $q(x)$. Les stratégies de clustering proposées dans ce mémoire vont nécessiter comme étape préalable une estimation de cette intensité q. Ce chapitre a pour but de présenter la méthode utilisée à cette fin.

Nous allons voir que nous pouvons déduire cette estimation d'intensité d'une estimation de densité. Cette dernière est présentée dans le cas univarié.

3.1 Estimation de la densité

Supposons $x_1, \ldots, x_n$, n observations d'une variable aléatoire de densité inconnue f, que nous cherchons à estimer.

Nous nous sommes concentré uniquement sur l'estimateur des noyaux. Cet estimateur est défini par

$$\hat{f}_h(x) = \frac{1}{n} \sum_{i=1}^{n} K_h(x - x_i),$$

où $K_h(x) = \frac{1}{h} K(\frac{x}{h})$.
La fonction $K(.)$ est appelée *noyau de l'estimateur*. Ce noyau représente la forme du poids mis sur chaque observation. Le *paramètre de lissage* h, comme son nom l'indique, caractérise le "lissage" de l'estimateur $\hat{f}_h$.

L'estimateur ainsi obtenu prendra une valeur élevée au voisinage de données denses, et des valeurs plus faibles où les données sont plus rares.

Le choix d'un estimateur de densité se ramène donc ici aux choix du noyau et du paramètre de lissage.

3.2 Choix du noyau

Le choix de ce noyau, comme l'a montré par exemple Silverman [16], n'est pas d'une grande importance. Dès lors, pour son implémentation aisée, nous avons choisi un *noyau uniforme*

$$K(x) = \begin{cases} 1 & \text{si } \|x\|_\infty \leq 1 \\ 0 & \text{sinon.} \end{cases}$$

Cependant, ce noyau s'est révélé dans certains cas inapproprié, associant dans l'estimateur une densité nulle à tout point situé à une distance supérieur à h d'une observation. Nous avons alors eu recours à un *noyau géométrique*, qui s'exprime différemment,

$$K(x, y, h) = \begin{cases} 1 - h & \text{si } x = y \\ \frac{1}{2}(1-h)h^{|x-y|} & \text{sinon,} \end{cases}$$

où $h \in (0, 1)$.

Nous conserverons cependant des notations adaptées au noyau uniforme. Le passage à la notation du noyau géométrique se fait immédiatement en remplaçant $K(\frac{x-y}{h})$ par $K(x, y, h)$.

3.3 Choix du paramètre de lissage

Celui-ci est par contre crucial. En effet, une valeur du paramètre de lissage proche de zéro aura pour effet un estimateur fort déchiqueté, car très sensible aux différentes observations. Inversément, de grandes valeurs de h tendront à produire un estimateur très lisse, mais qui risque de ne pas refléter les variations légères de la densité à estimer. Il convient donc de choisir un paramètre de lissage qui équilibre ces tendances.

Deux estimateurs rappelés par M.M. Brooks [8] ont retenu notre attention.

3.3.1 Kullback-Leibler

Il paraît clair que l'estimateur $\hat{f}_h$ cherché doit être le plus proche possible de la densité inconnue f. Il est donc raisonnable de chercher un paramètre de lissage qui minimise [1]

$$MISE(h) = E\left[\int (\hat{f}_h(x) - f(x))^2 dx\right]. \tag{3.1}$$

Pour cela, Kullback et Leibler proposent de choisir le h qui maximise

$$KL(h) = \prod_{i=1}^{n} \hat{f}_{h\,i}(x_i), \tag{3.2}$$

où $\hat{f}_{h\,i}(x)$ est l'estimateur du LOO de x (celui-ci est construit à partir de l'échantillon dont on a soustrait l'observation i)

$$\hat{f}_{h\,i}(x) = \left(\frac{1}{n-1}\right) \sum_{\substack{j=1 \\ j \neq i}}^{n} K\left(\frac{x - x_j}{h}\right). \tag{3.3}$$

3.3.2 Rudemo-Bowman (Least-squares cross-validation)

On cherche ici le h qui minimise non plus $MISE(h)$, mais [2]

$$\begin{aligned} ISE(h) &= \int [\hat{f}_h(x) - f(x)]^2 dx \\ &= \int \hat{f}_h(x)^2 dx - 2\int \hat{f}_h(x) f(x) dx + \int f(x)^2 dx. \end{aligned} \tag{3.4}$$

Ces auteurs proposent de choisir le h qui minimise

$$RB(h) = \int \hat{f}_h(x)^2 dx - \frac{2}{n} \sum_{i=1}^{n} \hat{f}_{h\,i}(x_i), \tag{3.5}$$

car $\int f(x)^2 dx$ ne dépend pas de h et $\frac{1}{n} \sum_{i=1}^{n} \hat{f}_{h\,i}(x_i)$ est un estimateur des moments de $\int \hat{f}_h(x) f(x) dx$. En effet,

$$E\left[\frac{1}{n} \sum_{i=1}^{n} \hat{f}_{hi}(x_i)\right] = E[\hat{f}_{hn}(x_n)] = E\left[\int \hat{f}_{hn}(x) f(x)\, dx\right] = E\left[\int \hat{f}(x) f(x)\, dx\right],$$

car $\hat{f}$, et donc $E(\hat{f})$, ne dépend que du noyau et du paramètre de lissage, et pas de la taille de l'échantillon.

---

[1] $MISE$ signifiant Mean Integrated Square Error.

[2] ISE signifiant Integrated Square Error.

3.3.3 Remarque

Pour n grand, $ISE(h) \approx MISE(h)$, mais, d'après M.M. Brooks [8], la minimisation de $RB(h)$ présente plusieurs avantages par rapport à la maximisation de $KL(h)$ (notamment moins de sources d'erreur). Cependant, pour son implémentation plus rapide et efficace sur les exemples, c'est l'estimateur de Kullback-Leibler que nous avons utilisé.

La table 3.1 reprend les valeurs du paramètre de lissage ainsi estimées, pour les exemples donnés en 1.3.

Exemple	h
r4.dat	10
r3.dat	10
clust_40.ex3	10
couronne.dat	20
couronne2.dat	20
noncv_36.dat	5
noncv_38.dat	5
r5.dat	10
r1.dat	10
cluster2.sas	14
clust_45.ex5	10
clust_44.ex4	5
clust_49.ex13	5

Figure 3.1: Estimation du paramètre de lissage pour quelques exemples.

3.4 Estimation de l'intensité

En ce sens que, tout comme pour la fonction de densité, nous nous attendons à avoir beaucoup d'observations lorsque la fonction d'intensité prend des valeurs élevées, et peu d'observations lorsque celle-ci prend des valeurs plus faibles, ces deux fonctions sont fort similaires. De ce point de vue, il apparaît naturel de considérer comme estimateur de l'intensité $q(x)$ l'estimateur des noyaux

$$\hat{q}(x) = \frac{1}{h} \sum_{i=1}^{n} K\left(\frac{x - x_i}{h}\right).$$

Au facteur multiplicatif de normalisation $\frac{1}{n}$ près, cette estimation est une estimation de densité. Ce facteur n'a cependant ici plus aucune raison d'être.

D'autre part, si nous utilisons un paramètre de lissage unique, le facteur $\frac{1}{h}$ devient également superflu. C'est le cas dans ce mémoire. L'estimateur d'intensité que nous considérons s'écrit donc simplement

$$\hat{q}(x) = \sum_{i=1}^{n} K\left(\frac{x - x_i}{h}\right), \tag{3.6}$$

où $K(\,.\,)$ est un noyau uniforme (ou géométrique, moyennant une réécriture de celui-ci), et h est obtenu par maximisation de $KL(h)$.

Chapitre 4

Optimisation d'une classification initiale

Le principe de cette approche est d'optimiser, selon notre critère, une classification initiale.

Rappelons que notre critère consiste à chercher une partition des points en m classes telle que l'intensité intégrée sur les enveloppes convexes des classes soit minimale. La règle d'affectation associée à ce critère consiste, face à une nouvelle observation, à affecter celle-ci à la classe pour laquelle l'intensité intégrée ajoutée à l'enveloppe convexe de la classe est minimale.

Nous savons que la méthode du K-means[1] ne fournit pas toujours de solution optimale. Elle a cependant l'avantage de produire des classes convexes, et peut donc servir de classification initiale pour notre approche.

Nous exposerons trois processus itératifs d'optimisation de la classification initiale, notée $\mathcal{C}_0$. Leur schéma d'exécution est identique :

1. Estimation de l'intensité $\hat{q}$ du processus de Poisson non homogène. Poser $i = 0$.
2. Considérer la partition $\mathcal{C}_i$.
3. Envisager la réaffectation de certains points. On obtient alors la partition $\mathcal{C}_{i+1}$.
4. Si $g(\mathcal{C}_{i+1}) < g(\mathcal{C}_i)$, aller en 2. Sinon, arrêter le processus.

[1] Pour une description détaillée de celle-ci, voir [6].

Rappelons que $g(\mathcal{C}) = \sum_{i=1}^{m} \int_{C_i} \hat{q}(x)dx$.

Un point changeant de classe uniquement si cela diminue la valeur du critère, la réaffectation de chacun des points entraîne forcément une diminution du critère, d'où

$$g(\mathcal{C}_{i+1}) \leq g(\mathcal{C}_i),\ i = 1, ..., m.$$

La situation $g(\mathcal{C}_{i+1}) = g(\mathcal{C}_i)$ correspond à un équilibre. La partition $\mathcal{C}_i$ envisagée n'est plus modifiée par le processus, qui est donc arrêté.

Les trois processus vont différer dans le choix des points pour lesquels nous envisageons une réaffectation, ce qui correspond à l'étape 3 de l'algorithme.

4.1 Echange de singletons entre classes

4.1.1 Principe

Pour ce processus, nous considérons comme points susceptibles de changer de classe les sommets des enveloppes convexes de toutes les classes. Il est vite apparu nécessaire d'ordonner l'ordre dans lequel ces points sont tour à tour considérés. Pour cela, nous considérons le point qui, sur tous les points non encore considérés, maximise l'intensité intégrée ajoutée à sa propre classe.

Le point considéré est affecté successivement à chaque classe, de manière à déterminer pour quelle enveloppe convexe de classe ce point minimise l'intensité ajoutée. S'il apparaît que le point doive changer de classe (i.e. la classe qui minimise l'intensité ajoutée n'est pas la classe d'origine de ce point)[2], nous recommençons le processus avec la nouvelle partition.

L'algorithme s'arrête lorsque, pour une partition courante, aucun des sommets des enveloppes convexes n'est changé de classe.

4.1.2 Implémentation

Le code du programme ECHANGE[3], qui implémente cette première méthode, est donné en Annexe.

[2]et que cette affectation n'entraîne pas d'intersection des nouvelles enveloppes convexes.

[3]Tous les programmes réalisés dans ce mémoire sont écrits en FORTRAN 77. Ils ont été exécutés sur station munie du système ULTRIX V4.3.

4.1.3 Résultats

Nous donnons pour chaque exemple la classification initiale (K-means), ainsi que le résultat de l'optimisation par le processus.

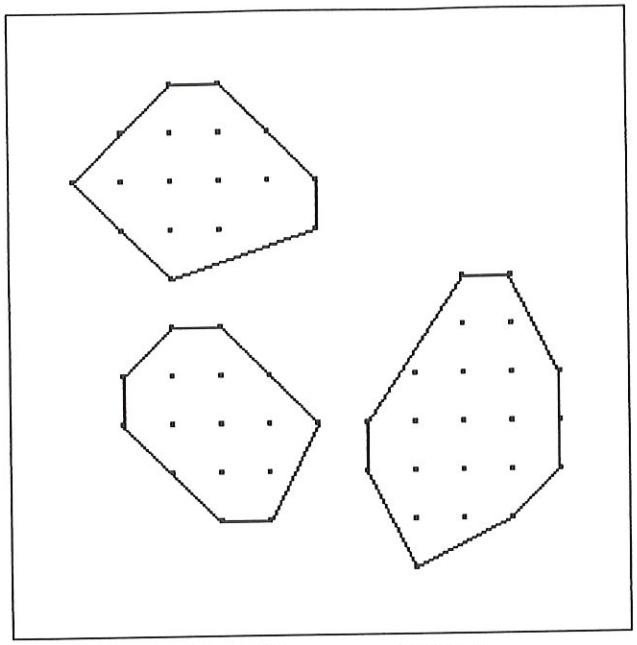
La valeur du critère correspondant à une classification est à chaque fois indiquée entre parenthèses.

Pour r1.dat, la solution qui résulte de l'optimisation est optimale. Nous remarquons qu'un seul point a changé de classe.

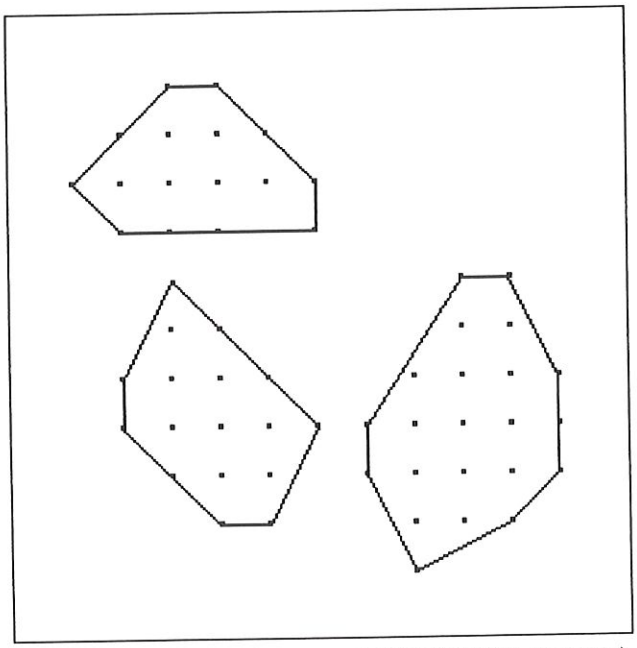
La classification obtenue par ECHANGE sur cluster2.sas est également optimale du point de vue du critère.

Pour r4.dat et r3.dat, même s'il y a eu une légère optimisation, nous sommes bien au-dessus de la solution optimale (pour laquelle la valeur du critère est 17280 pour r4.dat, et 16155 pour r3.dat). Les classifications initiales fournies par le K-means, composées de classes à tendance sphériques, sont en partie la cause de cela.

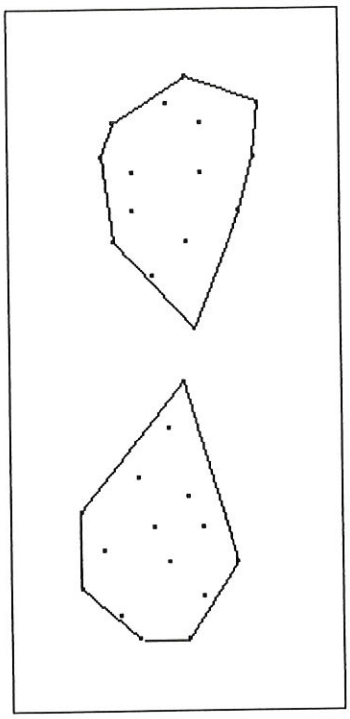
Pour clust_49.ex13, il y a une amélioration sensible, mais nous constatons que certains points, bien qu'éloignés, sont regroupés. Cela est dû au choix du noyau uniforme, qui a pour conséquence une intensité nulle dès que nous sortons de l'influence des points d'observation. Donc, si la zone située entre des points éloignés est pratiquement sans intensité, ECHANGE regroupera facilement ces points.



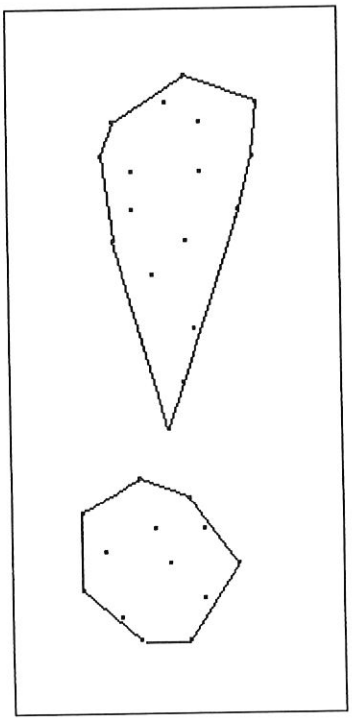
r1.dat : K-means (15654).



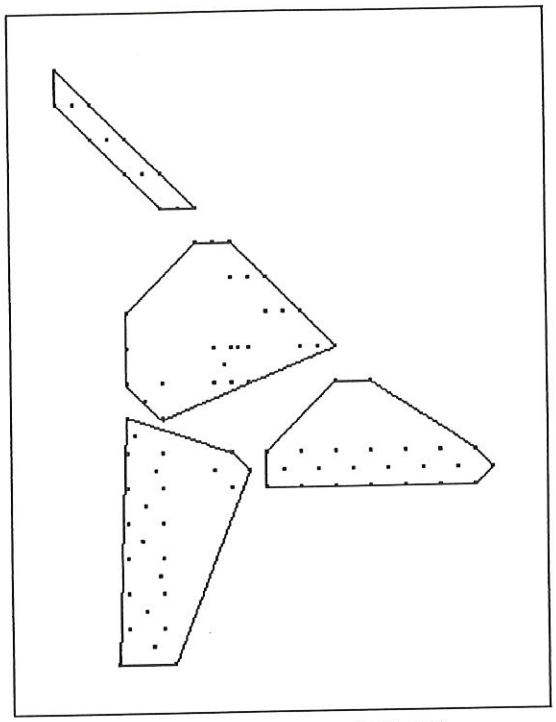
r1.dat : optimisation par ECHANGE (15471).



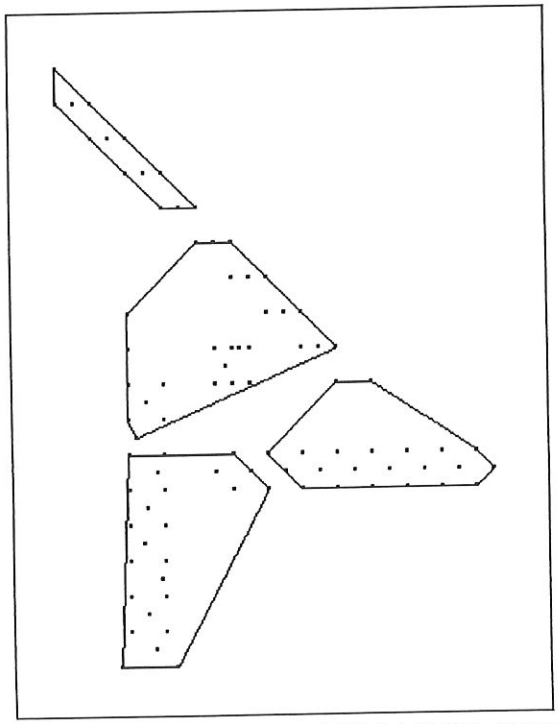
cluster2.sas : K-means (13854).



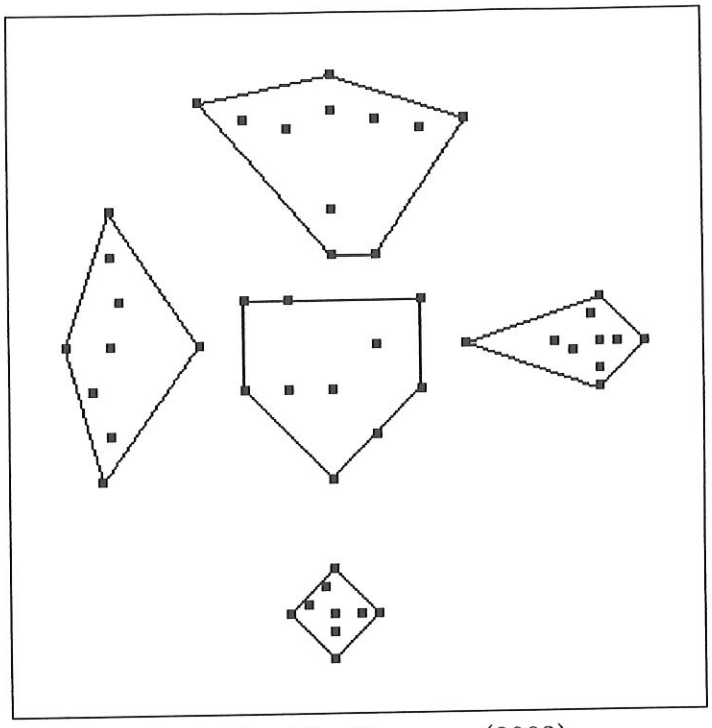
cluster2.sas : optimisation par ECHANGE (13194).



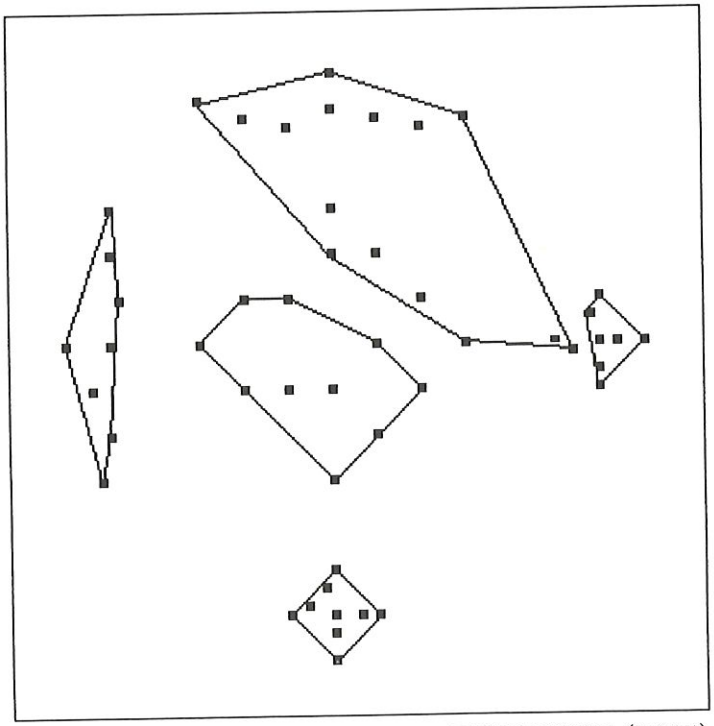
r4.dat : K-means (23788).



r4.dat : optimisation par ECHANGE (23579).



clust_49.ex13 : K-means (2993).



clust_49.ex13 : optimisation par ECHANGE (2645).

4.2 Echange de paires entre classes

Lorque deux sommets de l'enveloppe convexe d'une classe sont fort proches, et qu'ils devraient changer de classe[4], essayer d'en changer un seul à la fois (ce que fait le premier processus) ne marche pas, l'autre point contribuant à ce que l'intensité intégrée ajoutée à la classe d'origine du point soit faible. Cela apparaît clairement au vu des résultats du premier processus sur l'exemple r4.dat.

Une solution est donc d'envisager également le changement de classe de paires de points. Pour cela, nous maintenons l'ordre de considération des points, mais, alors qu'auparavant nous passions au point suivant si le point considéré ne changeait pas de classe, nous formons à présent une paire avec ce point, paire que nous affectons à la classe qui minimise l'intensité intégrée ajoutée par la paire. Si la paire change de classe, nous recommençons tout le processus. Sinon, nous envisageons la paire suivante formée par le point considéré. S'il n'y en a plus, nous considérons un nouveau point.

Le problème consiste à savoir comment choisir les paires associées à un point considéré. La première condition pour que deux points forment une paire est qu'ils appartiennent à la même classe.
Le premier point étant fixé, il paraît censé de choisir le second point parmi les sommets de l'enveloppe convexe résultant du retrait du premier point de sa classe.
Au sein des points envisageables pour compléter la paire, nous considérons d'abord, comme nous le faisions pour le premier point, celui pour laquelle l'intensité intégrée à l'enveloppe convexe ajoutée est maximale.

4.2.1 Implémentation

Le code d'ECHPAIRE n'est pas donné en Annexe. Si pour chaque point considéré, nous nous intéressons aux paires que l'on peut former à partir de celui-ci, nous obtenons ECHPAIRE à partir d'ECHANGE en choisissant le second point de le même manière que le premier.

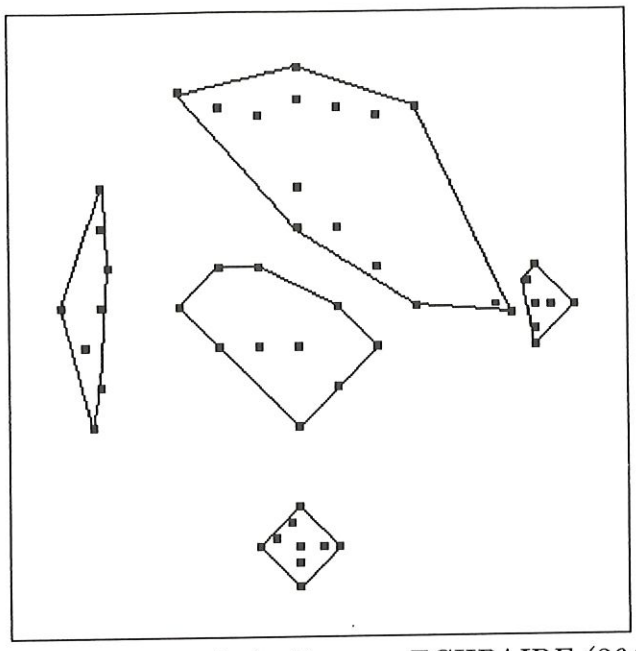
[4]C'est-à-dire que leur changement de classe conduirait à une partition plus optimale du point de vue du critère.

4.2.2 Résultats

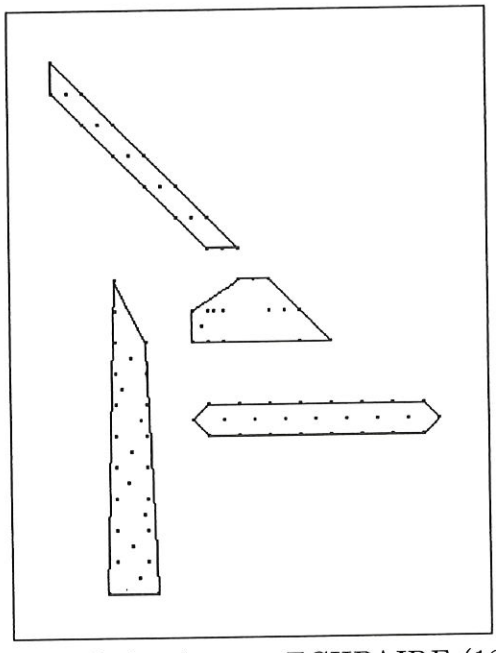
Il n'y a aucune amélioration par rapport à ECHANGE pour l'exemple r1.dat, ce qui est logique vu que la partition que nous avions obtenue était déjà optimale.

Il n'y a pas non plus d'amélioration par rapport au premier processus pour l'exemple clust_49.ex13. Nous nous attendions au changement de classe de la paire de points située à gauche de la classe de droite. Celui-ci n'a pas eu lieu, la zone ajoutée à l'enveloppe convexe de la classe supérieure étant de faible intensité.

Le cas de r4.dat est plus intéressant. Il y a ici une nette amélioration par rapport au premier processus, conformément à ce que nous espérions.



clust_49.ex13 : optimisation par ECHPAIRE (2645).



r4.dat : optimisation par ECHPAIRE (18422).

4.3 Restriction aux sous-enveloppes convexes

4.3.1 Principe

L'approche diffère ici des deux premiers processus. On a beaucoup étudié[5], face à l'enveloppe convexe d'une classe donnée, le degré de confiance que l'on a en la première sous-enveloppe, enveloppe convexe obtenue en retirant les sommets de l'enveloppe convexe de départ, voire même en la seconde sous-enveloppe.

L'idée est, à partir de la classification initiale, de considérer tour à tour chaque enveloppe convexe, de lui retirer ses sommets et de les réaffecter selon la règle de discrimination habituelle après avoir estimé l'intensité sans ces points.

Nous considérons en premier l'enveloppe convexe pour laquelle l'intensité intégrée ajoutée par les sommets est la plus grande, puis nous recommençons pour celles qui n'ont pas encore été considérées.

Une fois que l'enveloppe convexe à considérer est déterminée, nous lui retirons ses sommets, et estimons l'intensité sans ces points, étant considéré le fait que leur intensité ne peut être prise en compte pour leur affectation à une classe.

Ensuite, comme nous voulons conserver des enveloppes convexes disjointes, nous ne pouvons affecter indépendamment les points aux classes pour lesquelles ils minimisent l'intensité intégrée ajoutée. Nous affectons en premier le point qui ajoute l'intensité intégrée la plus faible à une classe, parmi tous les points à réaffecter et toutes les classes, et nous tenons compte des nouvelles enveloppes convexes pour les réaffectations suivantes, ainsi que de l'intensité qu'apporte le point nouvellement réaffecté.

Une itération se termine lorsque toutes les enveloppes convexes ont été considérées. A ce moment, comme indiqué en introduction au chapitre, si l'itération a entraîné une optimisation du critère, nous poursuivons le processus. Nous l'arrêtons dans le cas contraire.

Une amélioration est apportée à cette stratégie. Elle repose sur la considération suivante. Si, au cours d'une itération, il est plus intéressant de considérer plusieurs fois de suite les sous-enveloppes d'une même classe, il n'est pas logique de l'interdire, ce que nous faisions en considérant tour à tour chaque enveloppe une seule fois. La notion d'itération que nous utilisions n'a donc ici plus de sens.
Nous considérerons à chaque fois la sous-enveloppe convexe de la classe dont les sommets

[5]Voir travaux de Ruds et Rousseew, de Eddy B.

maximisent l'intensité intégrée ajoutée à la sous-enveloppe. Lorsqu'il n'y a plus de changement au sein d'une classe, nous empêchons que celle-ci soit à nouveau considérée tant qu'une autre classe n'a pas été modifiée.
L'implémentation de cette méthode est faite dans le programme SOUSENV.

Plutôt que de considérer successivement les enveloppes convexes des différentes classes, il est possible d'adopter une stratégie qui combine la première approche et les sous-enveloppes convexes.

Nous considérons les premières sous-enveloppes convexes de toutes les classes, ce qui revient à retirer tous les sommets des enveloppes de leur classe originale. Nous estimons alors l'intensité sans ces points, puis les réaffectons comme précédemment, c'est-à-dire aux classes qui minimisent l'intensité intégrée ajoutée par le point à l'enveloppe convexe de la classe, en commençant par le point qui minimise cela sur l'ensemble des points à réaffecter[6].

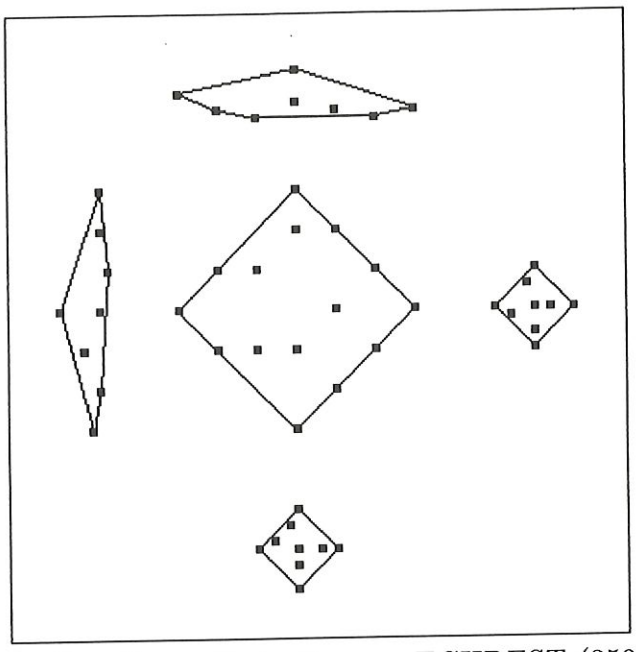
Cette modification a donné lieu au programme ECHREST, qui a été appliqué aux exemples clust_49.ex13 et r4.dat.

4.3.2 Résultats

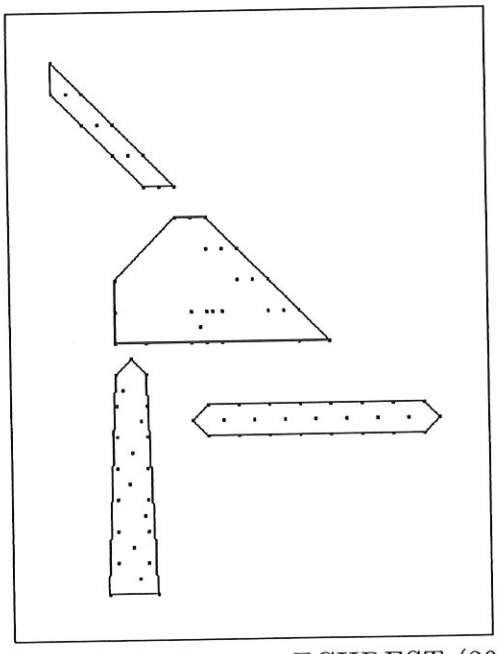
La solution obtenue pour clust_49.ex13 correspond à la solution préconisée.

Pour r4.dat, l'optimisation est un peu moins bonne qu'avec ECHPAIRE. Cela est dû au fait que nous commençons par réaffecter le sommet qui ajoute à une classe une intensité intégrée minimale. Ici, c'est le sommet situé à l'extrémité supérieure de la classe verticale, qui est affecté à la classe centrale, la zone le séparant de cette classe étant de faible intensité. Cela explique pourquoi le haut de la classe verticale est affecté à la classe centrale, ce qui ne correspond pas à une classification naturelle.

[6]Alors que la première approche affectait un point à la classe pour laquelle il minimisait l'intensité intégrée ajoutée à l'enveloppe convexe de la classe, nous nous intéressons ici à l'intensité intégrée ajoutée à la sous-enveloppe de cette classe.



clust_49.ex13 : optimisation par ECHREST (2501).



r4.dat : optimisation par ECHREST (20286).

4.4 Recours au noyau géométrique

4.4.1 Pertinence

Le choix du noyau uniforme a pour conséquence sur les exemples la présence de régions qui, étant sans observations, sont sans intensité. Notre critère, qui minimise l'intensité intégrée sur les enveloppes convexes des classes, aura donc tendance à inclure dans ces enveloppes des zones sans intensité, ce qui ne correspond pas à notre objectif. Implémenter les approches présentées dans ce chapitre avec un autre noyau, par exemple géométrique, devrait sensiblement améliorer les résultats de l'approche.

4.4.2 Application à ECHREST

Le programme ECHGEOM est l'analogue de ECHREST, mais l'estimation de l'intensité qui y est faite utilise un noyau géométrique et non uniforme. Le code d'ECHGEOM est donné en Annexe.

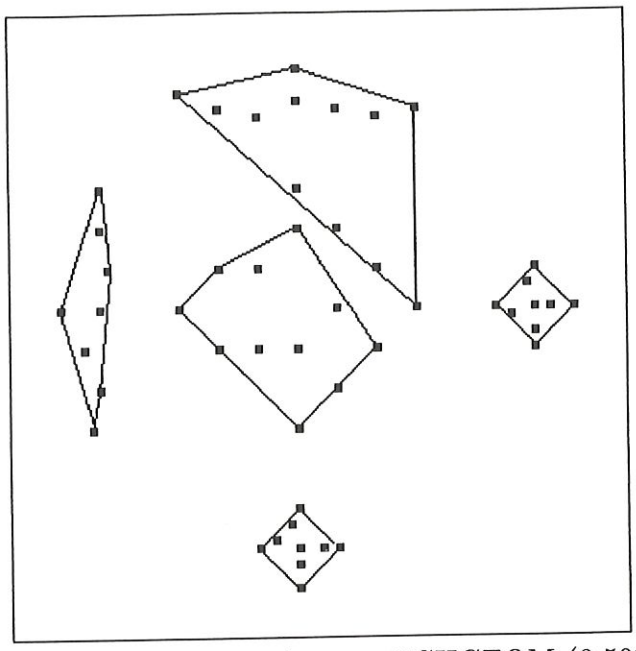
Appliquons ECHGEOM aux exemples r4.dat et clust_49.ex13.

La partition résultant du K-means conduit à une valeur du critère égale à 18,93266 pour r4.dat et 8,98308 pour clust_49.ex13.

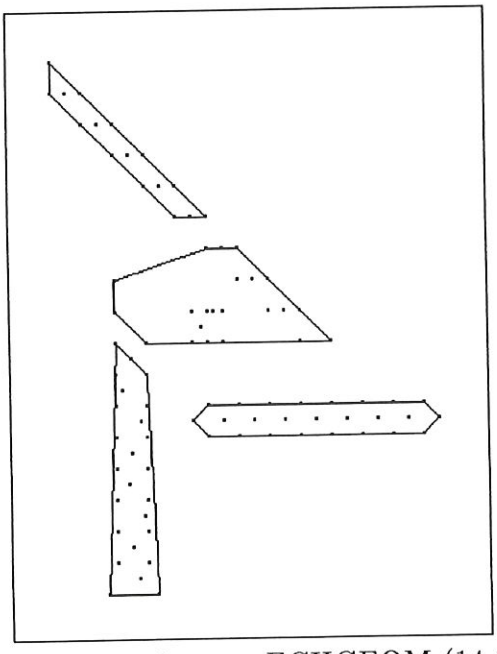
Notons que le h utilisé, qui est égal à 0,7, n'est probablement pas optimal. Il faudrait envisager la même estimation que pour le paramètre de lissage du noyau uniforme, par KL.

Sur r4.dat, la solution semble légèrement plus optimale que pour ECHREST, mais il doit être possible d'obtenir mieux.

Par contre, sur clust_49.ex13, l'optimisation est moins bonne que pour ECHREST.



clust_49.ex13 : optimisation par ECHGEOM (8,50246).



r4.dat : optimisation par ECHGEOM (14,14577).

Chapitre 5

Approche de la Connexité par la théorie des graphes

Il apparaît que l'hypothèse de convexité imposée pour avoir des classes acceptables est trop forte. Le but des deux chapitres suivants est de voir dans quelle mesure la convexité peut être remplacée par la connexité.

Il semble tout d'abord naturel de recourrir à des notions de théorie des graphes pour étudier cette connexité, les objets à classer étant assimilés aux sommets d'un graphe. C'est l'objet de ce chapitre.

Nous commencerons par rappeler ces notions, avant de présenter deux méthodes envisagées.

5.1 Notions théoriques

Définitions : On appelle

. *graphe* G, un couple $G = (\mathcal{O}, E)$, où chaque objet de $\mathcal{O}$ est appelé *sommet* et $E = \{(x_k, x_l) \text{ tq } x_k, x_l \in \mathcal{O},\ k \neq l\}$ est un ensemble d'*arêtes*.
On identifiera souvent par la suite graphe et ensemble de sommets.

. *sous-graphe* de G, un graphe $G' = (\mathcal{C}, E')$, où $\mathcal{C} \subseteq \mathcal{O}$ et $E = \{(x_k, x_l) \text{ tq } x_k, x_l \in \mathcal{C},\ k \neq l\}$.

. *chemin* dans un graphe entre deux sommets $x_k, x_l \in \mathcal{O}$, une séquence

$$((x_k, x_1), ..., (x_m, x_l))$$

où $x_i \in \mathcal{O}$, $i = 1, ..., m$ et chaque couple de la séquence est une arête du graphe.

. *composante connexe* d'un graphe G, un sous-graphe $G' = (\mathcal{C}, E')$ de G tel que, d'une part, pour tout couple de sommets de $\mathcal{C}$, il existe un chemin reliant ces sommets, et d'autre part $\mathcal{C}$ soit maximal par rapport à cette propriété.

Pour le problème qui nous concerne, un couple (x_k, x_l), $x_k, x_l \in \mathcal{O}$ constituera une arête du graphe si et seulement si

$$\|x_k - x_l\|_\infty \leq d,$$

où $d \in I\!R$.
Pour un ensemble de sommets fixés, nous appellerons *graphe de niveau d*, noté $G(d)$, le graphe résultant.

5.2 Recherche de partition acceptable

5.2.1 Critère

En clustering, nous voulons retrouver m domaines D_i disjoints et non vides. Pour cela, nous maximisons

$$\mathcal{L}(\mathcal{C}, h) = \prod_{k=1}^{n} f(x_i),$$

qui peut aussi s'écrire

$$\mathcal{L}(\mathcal{C}, h) = \prod_{i=1}^{m} \prod_{k=1}^{n_i} \frac{q_i(x_k, h)}{n_i}, \tag{5.1}$$

où n_i représente le nombre de points de la classe i, et

$$q_i(x, h) = \frac{1}{(2h+1)^2} \sum_{\substack{x_j \in D_i \\ x_j \neq x}} K(\frac{x_j - x}{h}),$$

où K est un noyau uniforme.

En prenant le logarithme dans (5.1), nous sommes amenés à maximiser

$$g(\mathcal{C}, h) = \sum_{i=1}^{m} \left[\sum_{x_k \in D_i} \ln q_i(x_k, h) - \ln(n_i) \right],$$

par rapport à $\mathcal{C}$ et h.

5.2.2 Principe de la méthode

Pour un h donné, cette approche consiste à passer en revue toutes les partitions *acceptables*, en gardant à chaque fois celle qui maximise le critère $g(\mathcal{C}, h)$.

Pour qu'une partition soit *acceptable*, il faut évidemment qu'aucune classe ne soit vide, mais surtout que l'ensemble des points forme un graphe de niveau h, les composantes connexes de ce graphe correspondant aux classes de la partition.

Problème : Il n'existe pas toujours de h qui fournisse ne fusse qu'une partition acceptable. En effet, si nous considérons l'exemple de la figure 5.1, pour lequel le plus petit h possible est 5 (un h plus petit ne conduit à aucun regroupement), nous constatons que tous les points sont associés à la même composante connexe.

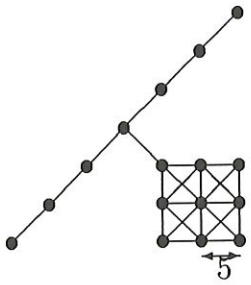


Figure 5.1: Graphe de niveau $h = 5$.

D'autre part, nous ne pouvons nous permettre de passer en revue toutes les partitions, même acceptables.

Ces raisons ont conduit à l'abandon rapide de cette méthode.

5.3 Recherche de parents dans un graphe

5.3.1 Principe de la méthode

Lorsque nous nous intéressions aux classes convexes, nous cherchions la partition qui minimise l'intensité intégrée sur les enveloppes convexes des classes.
Du point de vue de la connexité, une règle correspondante consisterait à chercher la partition en composantes connexes les plus "petites" possibles, c'est-à-dire d'intensité intégrée minimale.
Une approximation de cela peut se faire en considérant l'intégration sur un "couloir" entourant chaque arête reliant des sommets connexes.

La méthode consiste à associer un parent à chaque sommet.

Pour un h fixé (nous utilisons ici le même h que pour l'estimation de l'intensité q), nous considérons pour chaque point x_i le voisinage

$$V_i = \{j \text{ tq } \| \underbrace{x_i - x_j}_{\stackrel{not}{=} d_{ij}} \|_\infty \leq h\}, i = 1, ..., m\,.$$

L'idée de cette méthode est d'approximer l'intensité intégrée sur le "couloir" mentionné plus haut par la moyenne des intensités des deux observations situées aux extrémités de l'arête, cette moyenne étant pondérée par la distance (au sens de la norme infinie) entre ces sommets.
Nous calculons donc pour tout voisin de x_i cette intensité intégrée approximée, c'est-à-dire

$$\forall j \in V_i\ ,\ g_{ij} = \frac{q(x_i) + q(x_j)}{2}\ d_{ij}\ ,$$

où $q(x)$ est l'intensité de x.
Comme nous voulons minimiser l'intensité intégrée, nous choisissons comme parent de x_i le voisin x_j qui minimise g_{ij}, à condition que cette liaison entre x_i et x_j ne conduise pas à la formation de cycles dans le graphe.
Les composantes connexes du graphe résultant forment les classes finales.

Nous espérons que minimiser la somme des g_{ij} de cette manière revient bien à approximer la minimisation du critère rappelé en début de paragraphe.

La méthode est implémentée dans le programme CLUSTGRAPH.

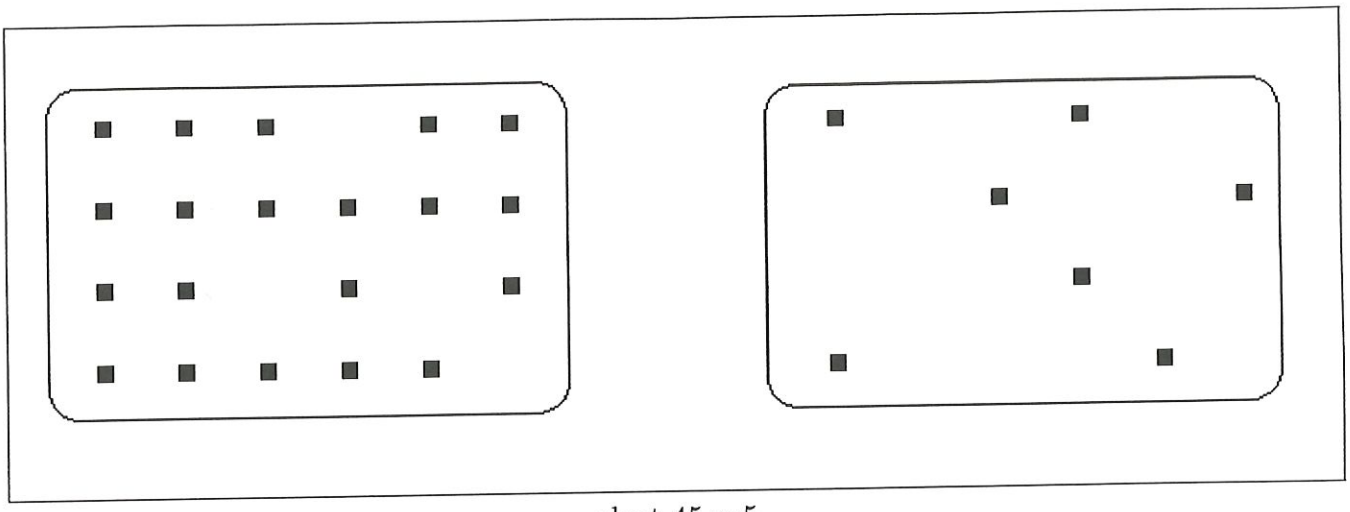
5.3.2 Résultats

La méthode fonctionne correctement sur les exemples clust_45.ex5, clust_49.ex13 et cluster2.sas.

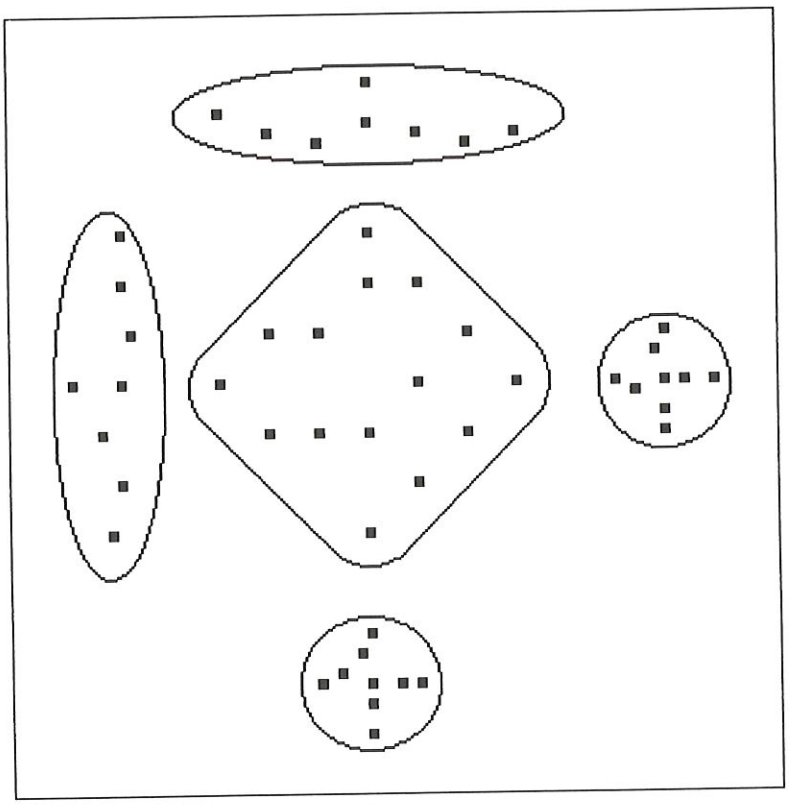
Cependant, et c'est le problème majeur, on ne peut influer sur le nombre de classes, qui résulte des parents que détermine CLUSTGRAPH. En conséquence, nous avons par exemple trois classes pour noncv_38.dat, au lieu des deux qui sont attendues.

Pour clust_40.ex3, le h minimum qui permet le regroupement de tous les points est égal à la distance entre les cinq classes attendues du bas de l'exemple. Il y a donc un risque de regroupement entre ces classes, et effectivement la méthode associe à des observations d'une classe attendue des parents n'appartenant pas à cette même classe.

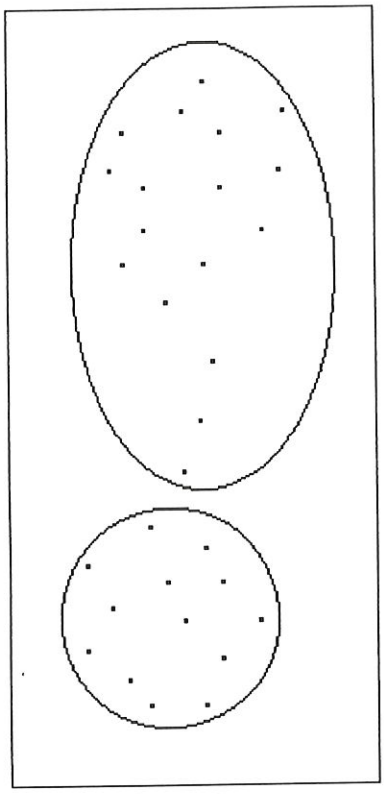
De même, pour couronne.dat, la méthode fournit quatre classes, au lieu de la couronne.



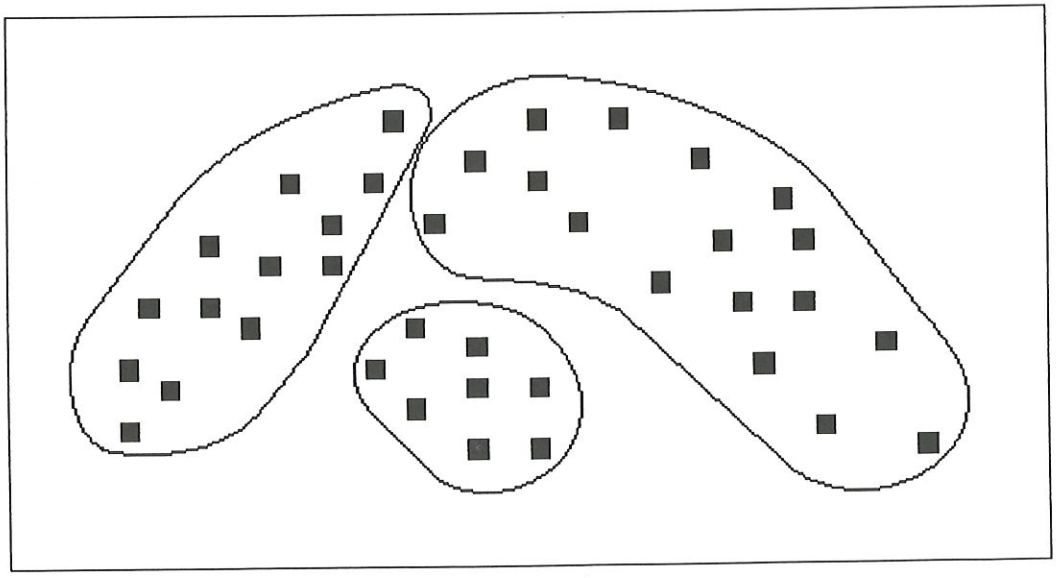
clust_45.ex5



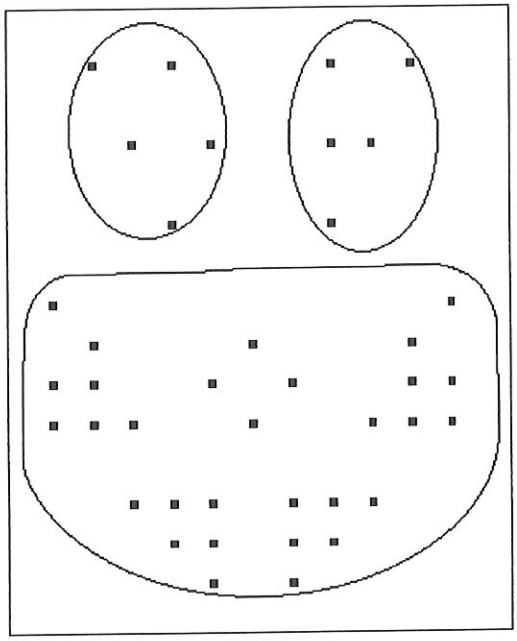
clust_49.ex13



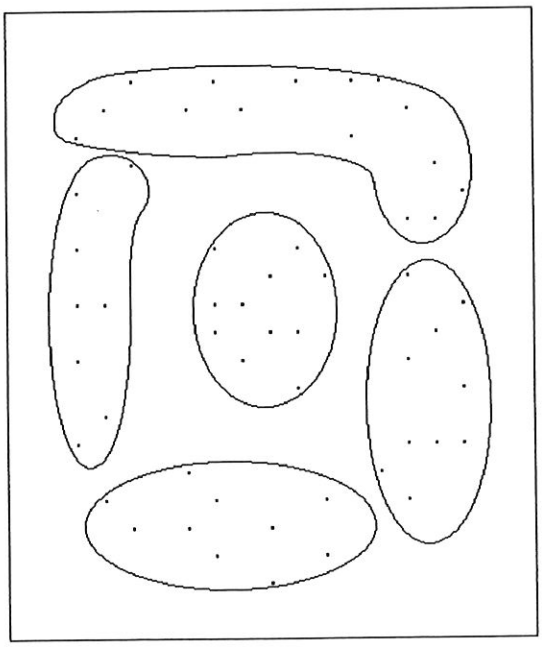
cluster2.sas



noncv_38.dat



clust_40.ex3



couronne.dat

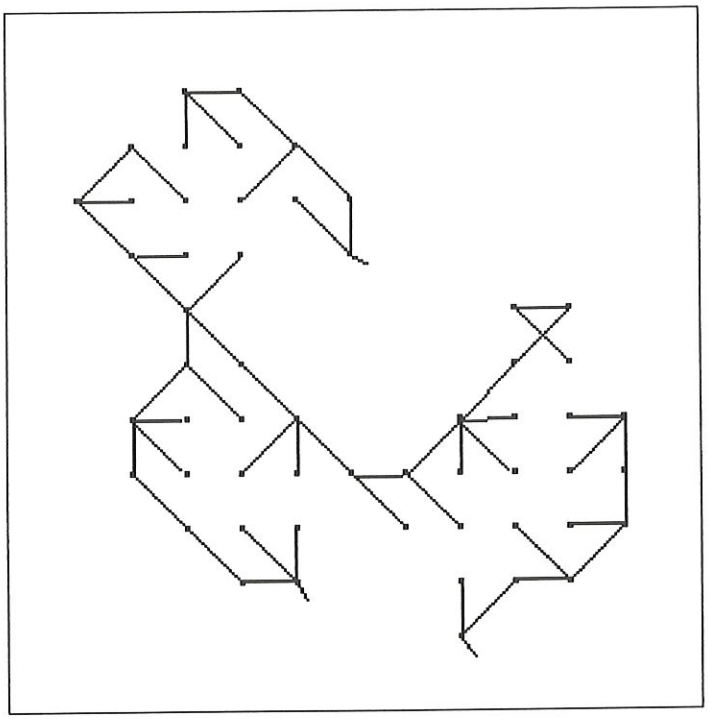
5.3.3 Etude plus détaillée de r1.dat

Sur cet exemple, CLUSTGRAPH fournit le nombre de classes attendu. Il est alors intéressant d'essayer de comparer la valeur du graphe construit par le programme, et celle d'un graphe qui correspondrait à la solution attendue, de manière à voir si l'approximation du critère ici utilisée contribue bien à donner la même solution optimale que le critère original en connexité.

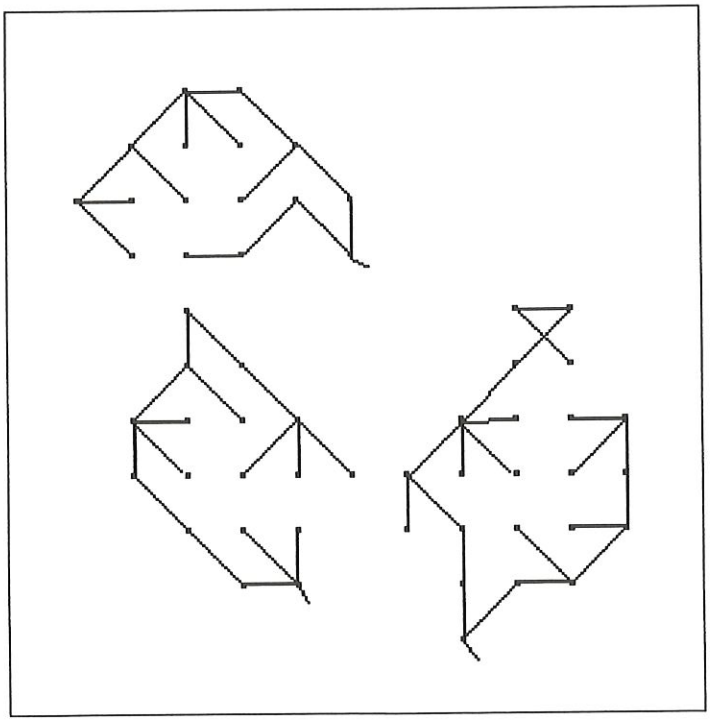
La somme des valeurs des arêtes du graphe correspondant à la solution attendue est 3085. C'est à mon avis la valeur minimale que nous puissions obtenir pour les composantes connexes désirées.

La valeur du critère pour le graphe construit par CLUSTGRAPH est cependant légèrement inférieure : 3075.

Cela signifie donc que l'optimisation de cette approximation du critère ne semble pas conduire à la même solution optimale que l'optimisation du critère original.



r1.dat : CLUSTGRAPH.



r1.dat : Solution attendue.

5.4 Conclusions

Notre nombre de classes étant fixé, envisager la méthode implémentée ici est délicat, vu qu'alors ce nombre de classes dépend de la recherche de parents. Cependant, si le nombre de classes obtenu par CLUSTGRAPH est supérieur au nombre attendu (comme c'est le cas pour les exemples clust_40.ex3 et couronne.dat), il est possible d'envisager un regroupement suivant le principe du plus proche voisin, au sens de cette connexité.

Remarquons que notre recherche de parents entraîne un problème de chaînage. Celui-ci provient probablement en partie du lien existant entre le paramètre de lissage utilisé pour le noyau uniforme, et le rayon des voisinages considérés.

Chapitre 6

Approche géométrique

6.1 Principe

Etant donné n points x_i, $i = 1, ..., n$, considérons autour de chaque point une "boule" centrée de rayon h_i, $\mathcal{B}(x_i, h_i)$. L'idée de base est de faire croître à chaque étape les boules

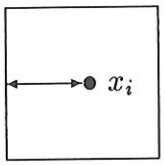


Figure 6.1: $\mathcal{B}(x_i, h_i)$.

autour de chaque point et de connecter deux points dès que les boules centrées en ces points s'intersectent. En effet, d'après Baddeley A.J. et van Lieshout M.N.M. [1], deux points sont "connexes" si les boules centrées en ces points s'intersectent. Nous espérons qu'en procédant ainsi les points seront progressivement connectés entre eux, jusqu'à former un nombre m de composantes connexes, m étant le nombre de classes attendu.

Le schéma ci-dessus montre clairement que nous travaillons avec des *cubes* et non des *boules* à proprement parler. Cependant, $\mathcal{B}(x_i, h_i)$ se caractérise mathématiquement par

$$\mathcal{B}(x_i, h_i) = \{x \text{ tq } \|x - x_i\|_\infty \leq h_i\},$$

c'est-à-dire l'ensemble des points situés à une distance inférieure ou égale à h_i du centre x_i, au sens de la norme infinie qui a été rappelée à la section 1.1.3. Cela justifie l'appellation de boule.

6.2 Augmentation du rayon des boules

h_i étant le rayon de la boule centrée en x_i pour une itération donnée, nous notons h_i^+ le rayon de cette boule à l'itération suivante. Le problème est de savoir quel accroissement donner à h_i, pour une itération quelconque.

Une première idée a été de faire croître toutes les boules de manière identique à chaque itération, sans aucun recours à quelque notion d'intensité que ce soit. Autrement dit,

$$h_i^+ = h_i + c \ , \ i = 1, ..., n \ ,$$

où c est une constante positive.
Cela donne des résultats comparables à ceux obtenus par la méthode du *Single linkage*.

Rappelons que notre critère de classification consiste à trouver les m groupes de points pour lesquels l'intensité intégrée sur les composantes connexes de ces groupes est minimale. Soit $\hat{q}$ l'intensité estimée du processus.
Pour rejoindre ce critère, l'idée a été de faire croître le rayon h_i autour de chaque point x_i de façon inversément proportionnelle à l'intensité intégrée sur $\mathcal{B}(x_i, h_i)$, c'est-à-dire

$$h_i^+ = h_i + \frac{1}{\int_{\mathcal{B}(x_i,h_i)} \hat{q}(x)dx} \ , \ i = 1, ...n \ .$$

Cela entraîne donc une augmentation plus rapide des rayons des boules centrées aux points ayant une intensité faible.
Cependant, en procédant de cette manière, on tient compte plusieurs fois des intensités des points situés aux intersections des boules, d'où une sur-estimation de l'intensité.

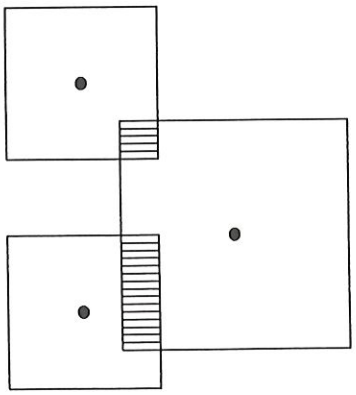


Figure 6.2: Intersection des boules.

Cette remarque nous a conduit à considérer l'augmentation suivante de h_i :

$$h_i^+ = h_i + \frac{1}{\int_{\tilde{\mathcal{B}}(x_i,h_i)} \hat{q}(x)dx} \ , \ i = 1, ...n \ .$$

où

$$\tilde{\mathcal{B}}(x_i, h_i) = \mathcal{B}(x_i, h_i) \setminus \{\mathcal{B}(x_j, h_j), j \neq i\}.$$

On ne tient donc plus compte des intersections des boules, ce qui implique une sous-estimation de l'intensité intégrée, mais moindre que la sur-estimation que nous avions.

Finalement, remarquons qu'il paraît inapproprié de tenir compte de l'intensité intégrée sur toute la boule $\tilde{\mathcal{B}}(x_i, h_i)$ pour l'augmentation de h_i. Il serait plus judicieux de ne tenir compte que de l'intensité intégrée sur une couronne $\tilde{\mathcal{C}}(x_i, h_i)$à la frontière de $\tilde{\mathcal{B}}(x_i, h_i)$, comme l'illustre le schéma de la figure ci-dessous.

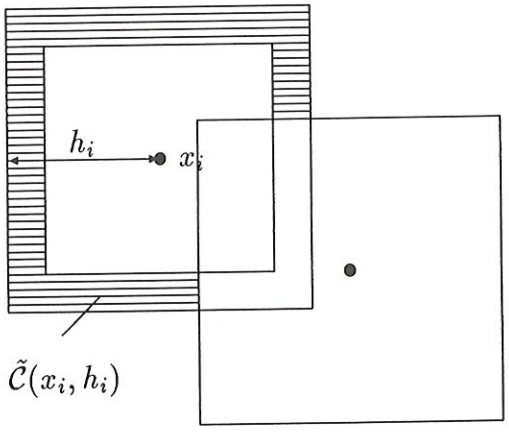


Figure 6.3: Modification de l'augmentation de h_i.

Une couronne de largeur constante égale à 1 a été utilisée.

Donc, à chaque étape, chaque point x_i $(i = 1, ..., n)$ voit le rayon h_i de sa boule $\tilde{\mathcal{B}}(x_i, h_i)$ augmenter de la manière

$$h_i^+ = h_i + \frac{1}{\int_{\tilde{\mathcal{C}}(x_i,h_i)} \hat{q}(x)dx} \ , \ i = 1, ...n \ .$$

6.3 Implémentation

Le code du programme CLUSTACC, qui met en pratique cette méthode, est donné en Annexe. En voici le principe.

Une fois que l'utilisateur a entré le nom du fichier de données (qui contient les coordonnées des points d'observation) et le nombre de classes voulu, la première étape est une estimation de l'intensité q.
Ensuite, tant que le nombre de classes voulu n'est pas atteint, nous augmentons simultanément le rayon de chaque boule comme vu en 6.2, et nous regardons quels points nous devons regrouper.

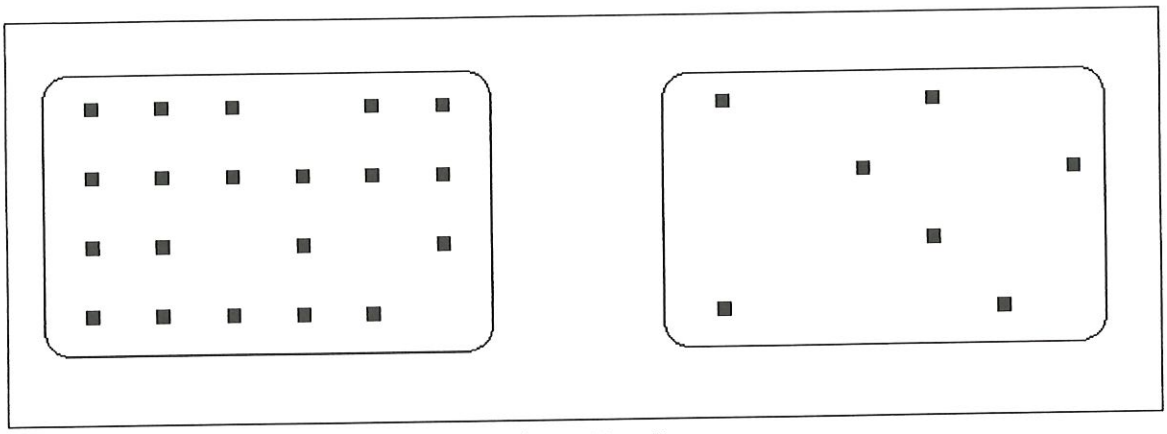
6.4 Résultats

La méthode semble bien fonctionner sur la plupart des exemples. Elle ne tend pas à produire un type particulier de classes, retrouvant aussi bien les classes sphériques de clust_45.ex5 que celles plus allongées de clust_44.ex4. Des classes non linéairement séparables comme celles de l'exemple r4.dat ne posent pas de problème. Il n'y a donc pas de problème pour r3.dat non plus.

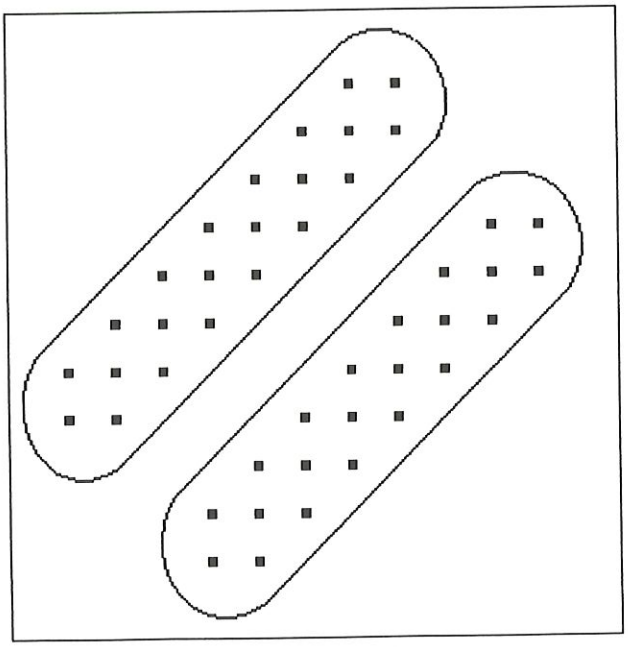
Son seul défaut semble être une tendance au chaînage. Cela se remarque déjà sur cluster2.sas. Cependant, nous avons vu au chapitre 4 que cette solution est celle qui minimise le critère utilisé, à savoir l'intensité intégrée sur l'enveloppe convexe des classes.

Les résultats obtenus pour couronne2.dat et noncv_38.dat sont plus gênants. En retirant les points qui semblent à l'origine du chaînage, c'est-à-dire en testant la méthode sur couronne.dat et noncv_36.dat, celle-ci se comporte de nouveau bien.

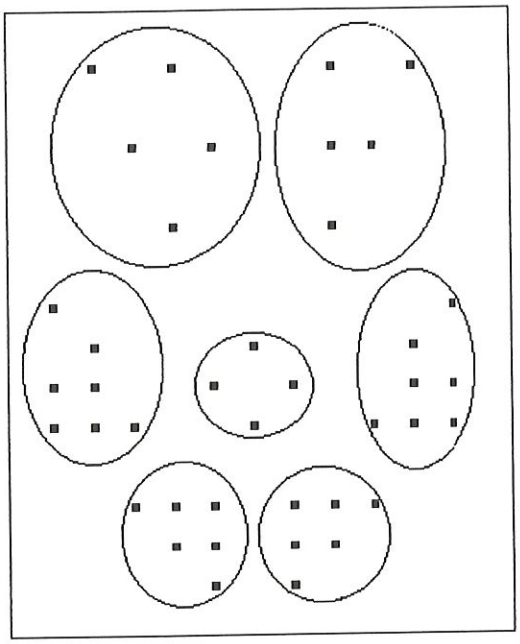
Le même type de problème survient avec l'exemple r1.dat. De plus, la forte symétrie de cet exemple semble renforcer la tendance à chaîner de la méthode. De nouveau, dès que l'on retire les deux points situés aux frontières des classes, ce qui revient à appliquer la méthode à r5.dat, il n'y a plus aucun problème.



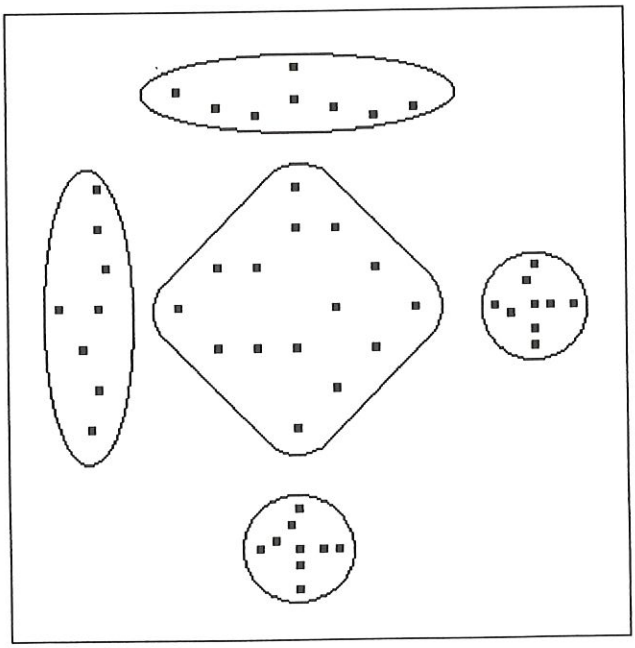
clust_45.ex5



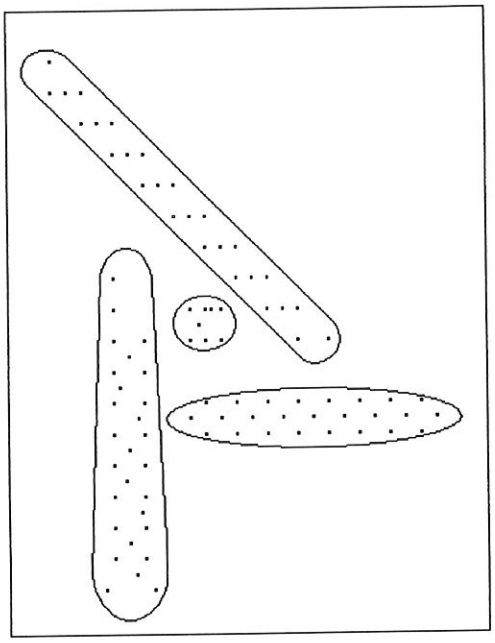
clust_44.ex4



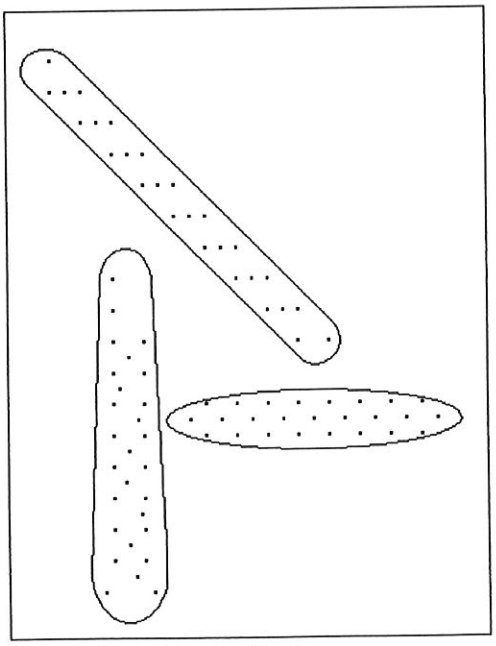
clust_40.ex3



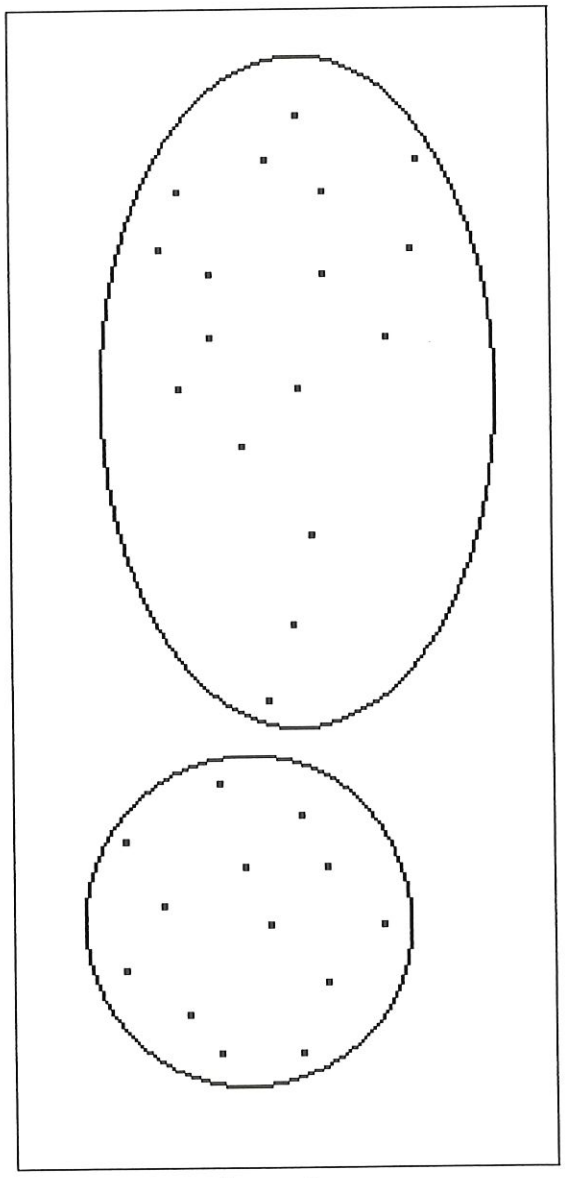
clust_49.ex13



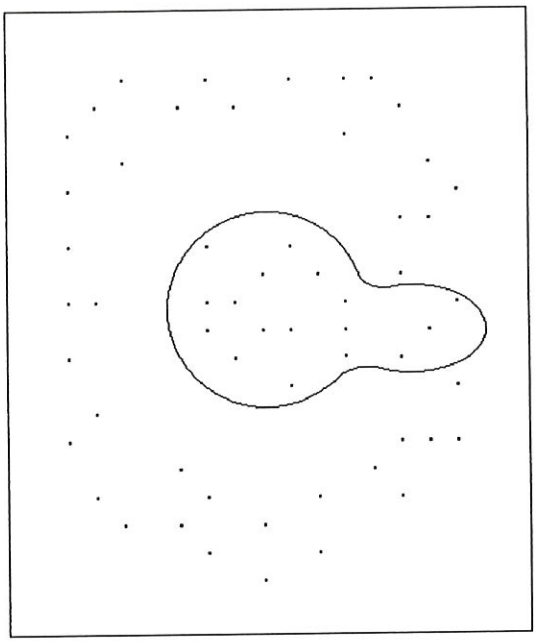
r4.dat



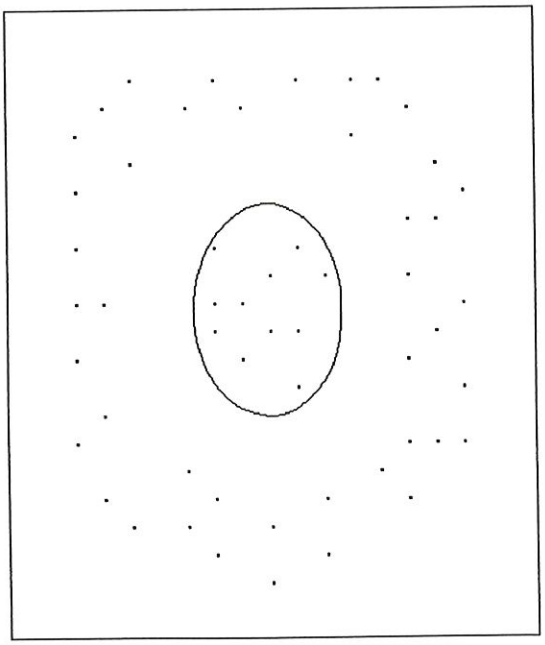
r3.dat



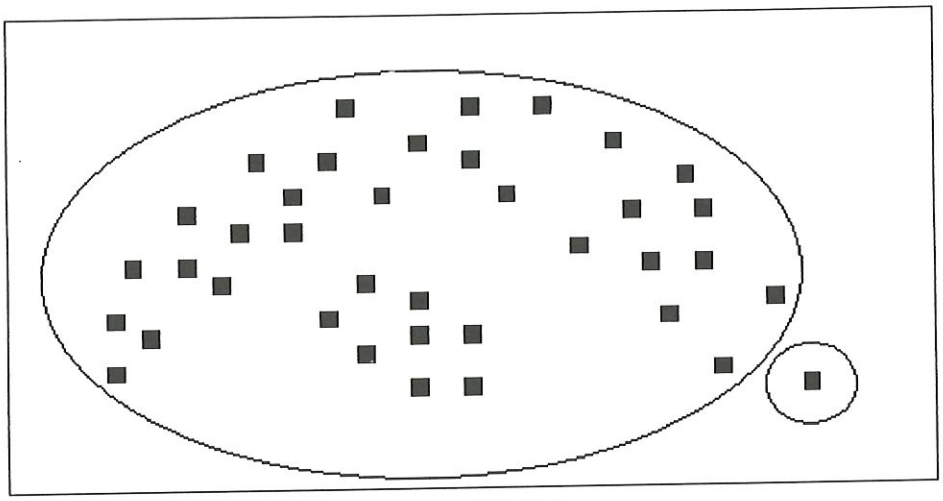
cluster2.sas



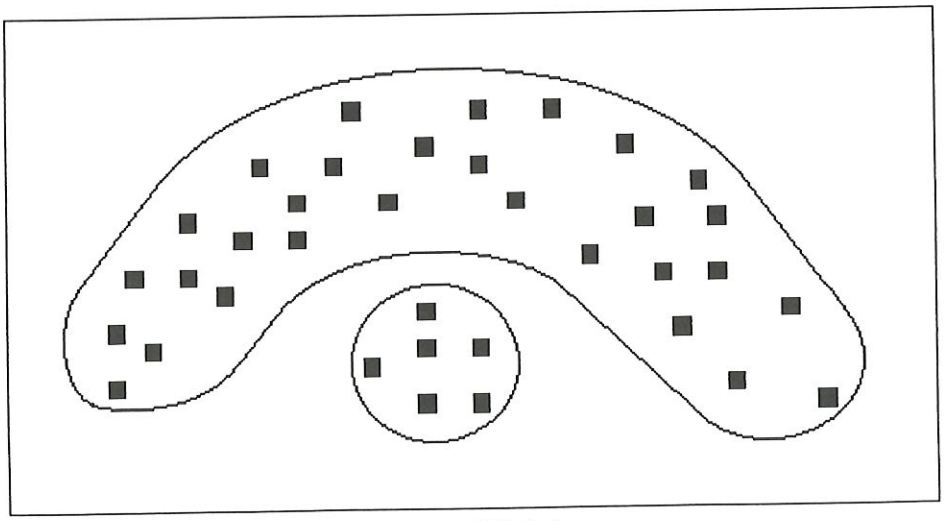
couronne2.dat



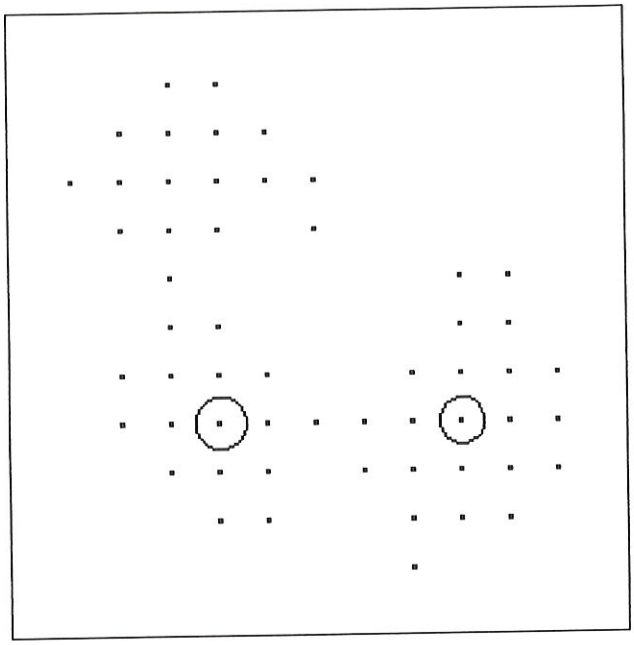
couronne.dat



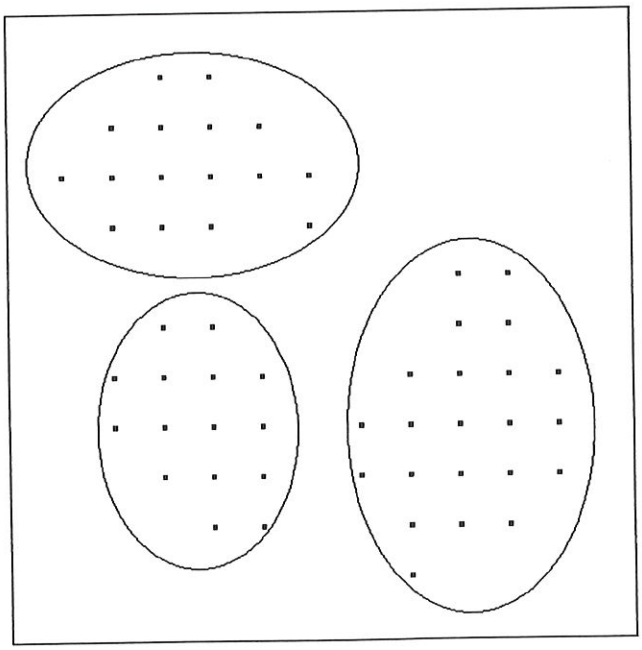
noncv_38.dat



noncv_36.dat



r1.dat



r5.dat

6.5 Modification de la relation de connexité

Notons que cette relation entre deux sommets était symétrique suivant ces sommets, vu que nous les connections lorsque les boules centrées en ces sommets s'intersectaient. Il est possible de modifier cette relation en supprimant la symétrie. Pour cela, nous cherchons un parent pour chaque sommet, en faisant croître autour de ce sommet une boule, comme c'était le cas auparavant. Seulement, c'est lorsque la boule d'un sommet recouvre un autre sommet que ce dernier devient parent du premier sommet.

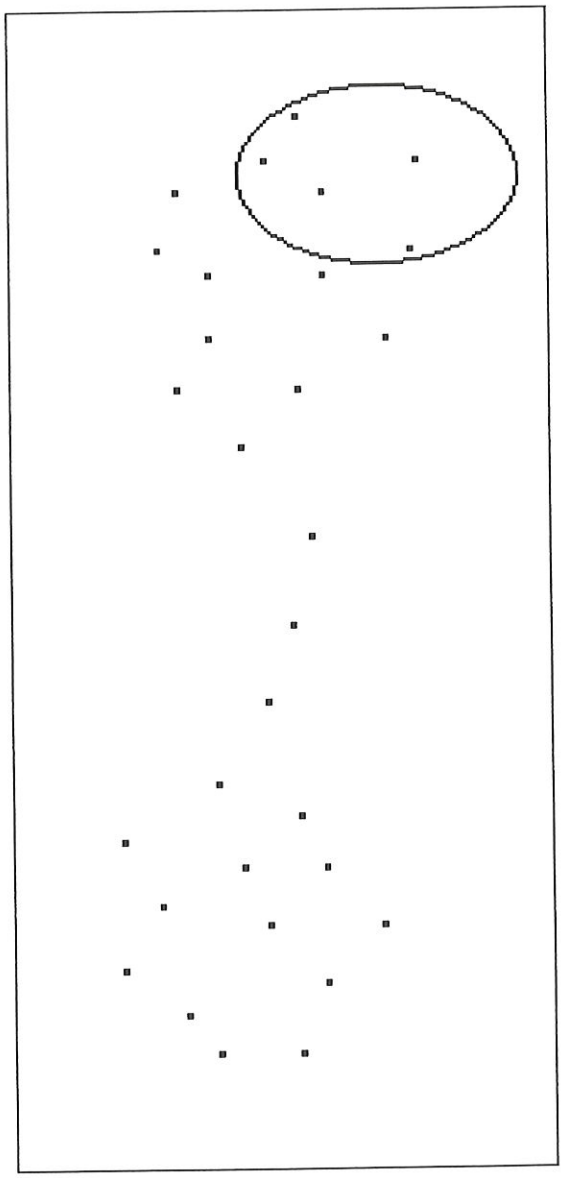
Cette remarque a donné lieu au programme NONSYM, qui est en fait identique à CLUSTACC, à ceci près que deux sommets sont regroupés seulement si une des boules centrées aux sommets recouvre l'autre sommet.

NONSYM a été testé sur divers exemples. Son comportement est identique à celui de CLUSTACC, excepté pour cluster2.sas, r1.dat et noncv_38.dat, qui sont repris ici.

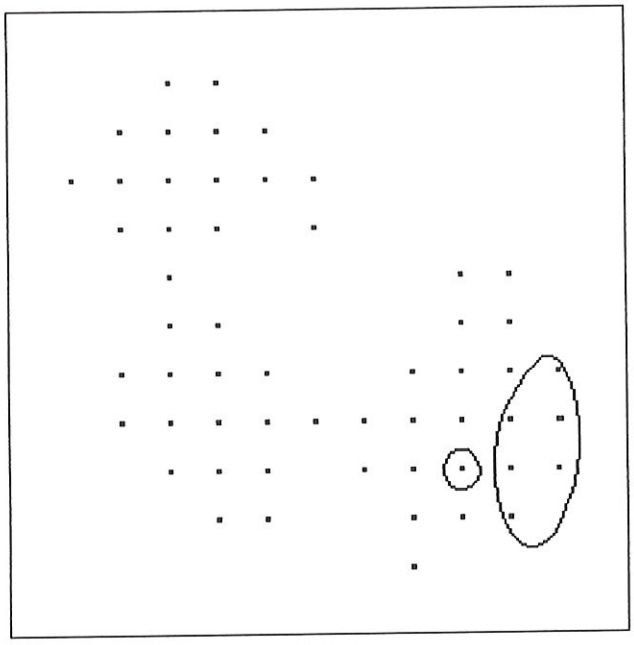
Pour cluster2.sas, le chaînage est accentué par rapport aux résultats obtenus avec CLUSTACC.

Les résultats obtenus pour r1.dat ne sont pas meilleurs que ceux obtenus avec CLUSTACC.

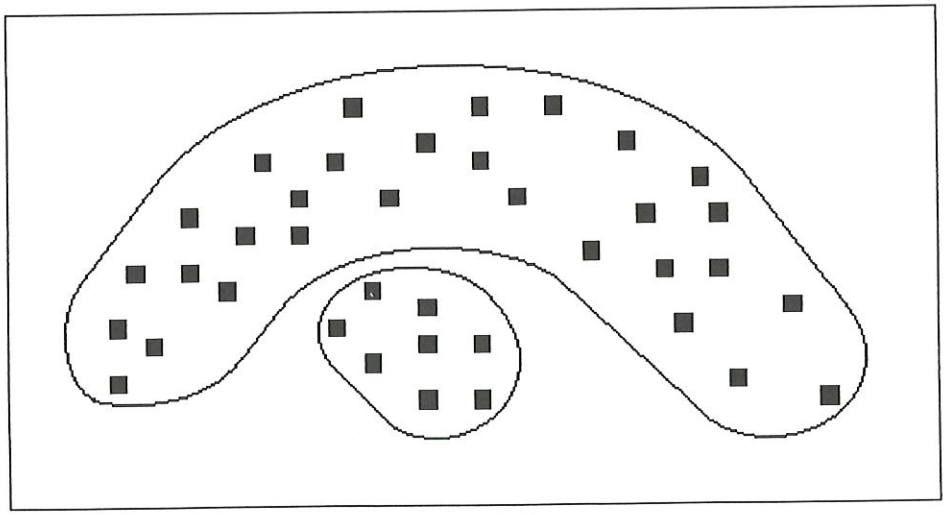
Par contre, NONSYM retrouve les deux classes de noncv_38.dat, ce que ne faisait pas CLUSTACC.



cluster2.sas



r1.dat



noncv_38.dat

6.6 Conclusions

Notons tout d'abord que dans cette approche, il existe un lien entre la taille de la boule d'estimation d'intensité et celle de la boule de connexité, tout comme au chapitre précédent il y en avait un entre la taille de la boule d'estimation et la taille des voisinages considérés.

L'inconvénient majeur de ce type d'approche reste le chaînage qui survient lorsque nous sommes en présence de ponts entre les classes. Cependant, il semble que ce soit le seul handicap dont souffre l'approche.

Chapitre 7

Discussion relative au choix du paramètre de lissage

Comme cela a déjà été dit au chapitre 3, le choix du paramètre de lissage pour l'estimation de l'intensité est très important. Certains choix semblent meilleurs que d'autres, mais qu'en est-il réellement ?

Les stratégies du clustering et de l'analyse discriminante sont fort proches. Pourquoi dès lors ne pas essayer d'utiliser des paramètres de lissage provenant de l'analyse discriminante dans notre problème de clustering ? Si les résultats s'avéraient être intéressants, il ne resterait plus ensuite qu'à arriver à retrouver ces paramètres de lissage sans avoir recours à l'analyse discriminante.

7.1 Estimation du paramètre de lissage en analyse discriminante par Leaving-One-Out

Comme base d'entraînement d'un exemple donné, nous considérons les classes formées par la solution attendue.

Connaissant donc la classe d'appartenance de chacun des points d'observation, nous choisissons le paramètre de lissage h (il en existe souvent plusieurs) qui maximise le nombre de points bien reclassés par la règle d'affectation ci-dessous

Un point d'observation x est affecté à la classe C_j $(j = 1, ..., m)$ si

$$j = \text{argmax}_{i=1,...,m} \left[\frac{1}{(2h+1)^2} \sharp \{x_l \in C_i (l = 1, ..., n,\ x_l \neq x) \text{ tq } \|x_l - x\|_\infty \leq h\} \right]. \quad (7.1)$$

Si C_j est la "vraie" classe de x, x est dit *bien reclassé.*

En fait, en considérant un noyau $K(\,.\,)$ uniforme, nous pouvons réécrire l'équation (7.1) comme

$$j = \text{argmax}_{i=1,...,m} \underbrace{\left[\frac{1}{(2h+1)^2} \sum_{\substack{x_l \in C_i \\ x_l \neq x}} K\left(\frac{x_l - x}{h}\right) \right]}_{\stackrel{not}{=} \hat{f}_i(x,h)}.$$

Notons j la "vraie" classe de x_j. Nous obtenons alors la règle suivante,

$$\text{Si } \max_{i=1,..,m} \hat{f}_i(x,h) = \hat{f}_j(x,h),$$
$$x \text{ est bien reclassé.}$$

Nous appellons *intervalle de bon reclassement maximal* l'intervalle contenant les h pour lesquels le nombre de points bien reclassés est maximal.

Sur tous les exemples testés, le paramètre de lissage obtenu en clustering par la maximisation de $KL(h)$ (voir chapitre 3) correspond à la borne inférieure de cet intervalle de bon reclassement maximal. Nous pouvons le voir dans la table 7.1 qui donne pour divers exemples l'intervalle de bon reclassement maximal ainsi que l'estimation obtenue en clustering.
Lorsque rien n'est spécifié, cela signifie que les h de l'intervalle de bon reclassement maximal reclassent correctement tous les points. Dans le cas contraire, le nombre de points mal classés (m.c.) est indiqué entre parenthèses.

Exemple	KL en Clustering	LOO en A.D.
r4.dat	10	[10,16]
r3.dat	10	[10,16]
clust_40.ex3	10	[10,14]
couronne.dat	20	[20,39]
couronne2.dat	20	[20,39]
noncv_36.dat	5	[5,6]
noncv_38.dat	5	[5,6]
r5.dat	10	[10,39]
r1.dat	10	[10,39] (1 m.c.)
cluster2.sas	14	[14,17]
clust_45.ex5	10	[10,24]
clust_44.ex4	5	[5,29]
clust_49.ex13	5	[5,9]

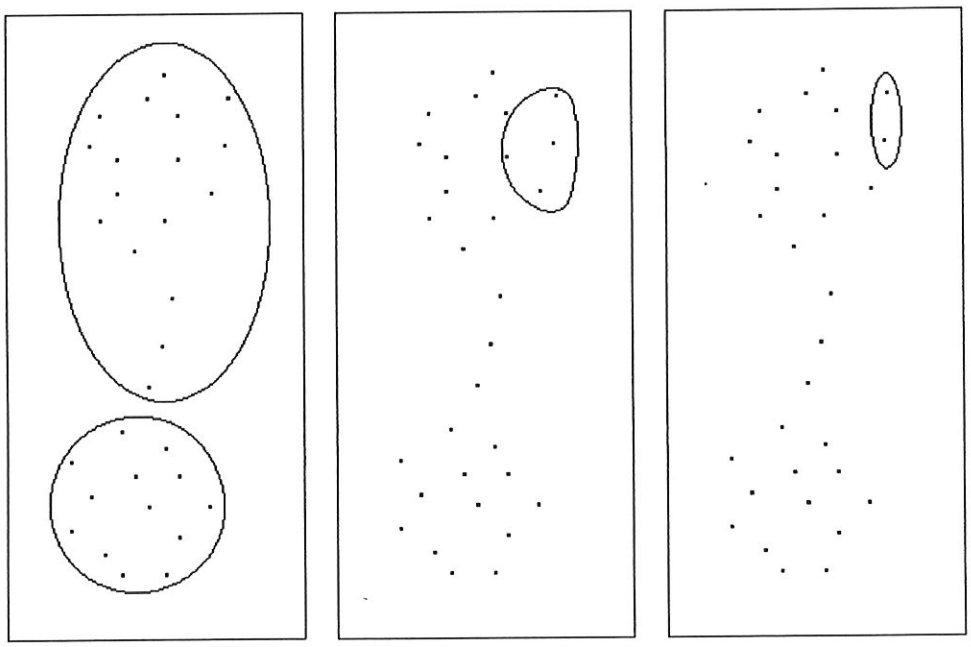
Figure 7.1: Estimation comparée du paramètre de lissage en analyse discriminante et en clustering.

7.2 Utilisation des h de l'analyse discriminante pour l'estimation de l'intensité en clustering

Considérons l'approche géométrique exposée au chapitre précédent, implémentée par CLUSTACC. Celle-ci a été testée pour divers valeurs de h sur plusieurs exemples. Les résultats pour les exemples cluster2.sas et r1.dat sont reproduits ici.

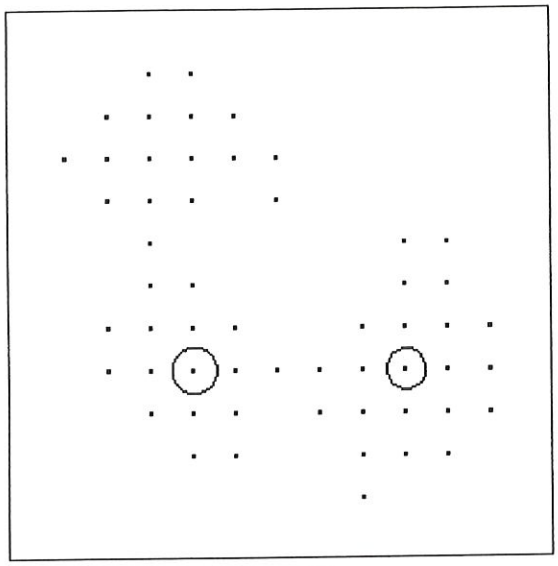
Tout comme c'est le cas pour cluster2.sas, nous avons remarqué que, pour tous les exemples, lorsqu'il existe un h pour lequel l'approche géométrique donne la solution attendue, c'est également le cas pour la borne inférieure de l'intervalle de bon reclassement maximal.

Cependant, nous constatons sur l'exemple r1.dat qu'aucun h ne fournit la partition attendue, mais que certains h apparaissent meilleurs que la borne inférieure de l'intervalle de bon reclassement maximal.

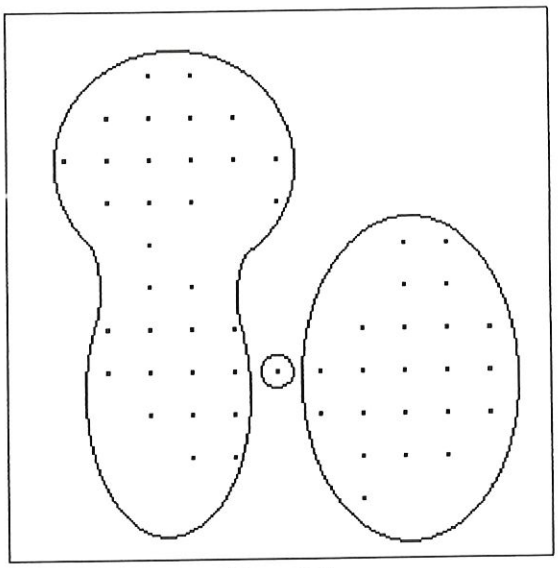


($h = 14$) ($h = 25$) ($h = 37$)

cluster2.sas



$(h = 10)$



$(h = 39)$

r1.dat

Conclusion

L'approche relative à l'optimisation de la classification initiale semble fort prometteuse, mais certaines remarques importantes doivent être faites. Tout d'abord, le choix de la classification initiale doit être amélioré. En effet, la tendance de la méthode du K-means à produire des classes sphériques handicape fortement l'approche, comme nous avons pu nous en rendre compte notamment sur l'exemple r4.dat. Ce défaut a déjà été mis en évidence, notamment par Kim Y.H., Lee J.C. et Lee S. [11], qui parlent de "groupements non naturels" réalisés par le K-means.
D'autre part, des inconvénients liés au nombre relativement faible de points doivent également être soulignés. En considérant des sous-enveloppes convexes, il arrive sur plusieurs exemples que le "noyau" qui subsiste ne soit plus réellement significatif, en raison du petit nombre de points qu'il contient. Une alternative serait de considérer, plutôt que les sous-enveloppes, l'enveloppe convexe résultant de la projection de l'intensité tronquée. Le niveau de cette troncature doit être choisi pour chaque classe. Pour ce faire, une possibilité est de rechercher les "modes" de l'intensité, diminuer celle-ci jusqu'à obtenir un noyau de taille suffisante, puis tronquer l'intensité. L'annexe A donne une méthode de détection de modes, et une idée d'algorithme dans lequel incorporer cette détection. Cette approche doit être envisagée.
Le noyau géométrique, qui semble plus efficace que le noyau uniforme pour ce type d'approche, doit être utilisé à l'avenir beaucoup plus intensivement, à condition que la sensible augmentation du temps calcul qui en résulte ne soit pas un handicap trop important.

Notons que dans une même perspective d'optimisation d'une classification initiale, il serait intéressant de considérer l'algorithme proposé par Kim Y.H., Lee J.C. et Lee S. dans [11], qui effectue une classification de points en regroupant les points "ambigus" dans une classe. Il reste ensuite à envisager l'affectation de ces points "ambigus" à l'une des autres classes, par notre règle de discrimination par exemple.

Le chaînage semble être le principal inconvénient des approches basées sur la connexité. Celui-ci a plusieurs explications. Tout d'abord, nous avons remarqué qu'aussi bien pour l'approche basée sur la théorie des graphes que pour l'approche géométrique, il y a un lien entre le paramètre de lissage utilisé pour l'estimation de l'intensité et la taille des voisinages pour la première approche, ou celle des boules de connexité de l'approche géométrique. Une

alternative consisterait à supprimer ce lien en considérant un autre noyau que l'uniforme. Le noyau géométrique doit convenir.
Cependant, ce chaînage provient également du fait qu'une composante connexe manque de structure interne, de "contenu" géométrique. Alors que la convexité semblait trop forte, la connexité souffre du problème opposé.
Il nous paraît intéressant d'envisager la combinaison des techniques de connexité et de convexité. La première est trop faible, pose problème en présence de ponts entre classes, la seconde est trop forte et incalculable. Il doit être possible de regrouper les points par connexité dans un premier temps, puis de considérer ensuite la réaffectation de certains de ces points par des techniques convexes.

Il nous semble important de mentionner que l'approche géométrique a été adaptée à des photographies de cellules en biologie, qui contiennent trois classes, et deux canaux spectraux. Le programme CLUSTACC, qui classait selon cette approche des points situés dans un espace spatial à deux dimensions, a été modifié de manière à classer des points, correspondant aux pixels[1], à deux dimensions spectrales.
Cela s'avère efficace lorsque le nombre de pixels est faible, i.e. sur les plus petites images (5×5 ou 15×15 pixels). Cependant, lorsque ce nombre de pixels augmente, la proximité spectrale entre les pixels provenant de classes attendues différentes augmente également, la frontière entre les classes étant floue sur les photographies. Par conséquent, l'approche entraîne rapidement un important chaînage. Il faudrait introduire une contrainte spatiale, ou combiner clustering spectral et clustering spatial.

L'estimation du paramètre de lissage en recourrant à l'analyse discriminante appelle également certains commentaires.
Rappelons tout d'abord ce qui a été constaté au dernier chapitre, à savoir que l'estimation par KL en clustering donne systématiquement sur les exemples une borne inférieure de l'intervalle de bon reclassement maximal obtenu en analyse discriminante par Leaving-One-Out. Si nous nous intéressons à cette borne inférieure, le recours à l'analyse discriminante devient donc superflu, ce qui est fort important. Or, nous avons également remarqué que, lorsqu'un h fournit la classification attendue, alors c'est aussi le cas pour celui obtenu par KL. Cela semble donc plaider en faveur du choix de cette borne inférieure pour l'estimation du paramètre de lissage. Remarquons cependant que, lorsqu'il n'y a pas de h qui conduit à cette classification attendue, alors le h obtenu par KL n'est pas forcément le plus intéressant.
Il serait important d'essayer de formaliser ces remarques.

[1] *Pixel*, terme anglo-saxon provenant de *Picture Element*, peut être traduit par *point image*.

Bibliographie

[1] Baddeley A.J., van Lieshout M.N.M. (1993), *Stochastic geometry models in high-level vision*, in : Statistics and images, Mardia K., Kanji G.K. (Eds), Vol. 1 of Advances in Applied Statistics, a supplement to Journal of Applied Statistics 20, pp. 231-256.

[2] Beaufays P., Rasson J.-P. (1984), *Une nouvelle règle de classement, utilisant l'enveloppe convexe et la mesure de Lebesgue*, Statistique et Analyse des Données, 2, pp. 31-47.

[3] Beaufays P., Rasson J.-P. (1984), *Propriétés théoriques et pratiques et applications d'une nouvelle règle de classement*, Statistique et Analyse des Données, vol. 9/3, pp. 1-10.

[4] Beaufays P., Rasson J.-P. (1985), *A new geometric discriminant rule*, Computational Statistics Quaterly, 2, pp. 15-30.

[5] Bertholet V., Boudart G., Lissoir S. (1995), *Vers une automatisation des classifications sur base des techniques de convexité*, Mémoire de Licence, F.U.N.D.P. Namur, Belgique, inédit.

[6] Bock H.H., *Cours de Classification*, Département de mathématiques F.U.N.D.P. Namur, Belgique. Année académique 1994-1995.

[7] Bock H.H. (1995), *Tests de validation d'une classification*, Session plénière, Troisièmes Journées de la S.F.C., 28-29 septembre 1995, Namur, Belgique.

[8] Brooks M.M. (1991), *Bandwidth selection methods for kernel estimators of the intensity function of a nonhomogeneous Poisson process*, Phd. Thesis, University of North Carolina, Chapel Hill.

[9] Gordon A.D. (1981), *Classification*, Chapman and Hall, London.

[10] Hardy A., Rasson J.-P. (1982), *Une nouvelle approche des problèmes de classification automatique*, Statistiques et Analyse des Données, 7, pp. 41-56.

[11] Kim Y.H., Lee J.C., Lee S. (1996),*A clustering method with an ambiguous class*, Abstract, Lecture in the fifth conference of I.F.C.S., March 27-30, 1996, Kobe, Japan.

[12] Koontz W., Narendra P.M., Fukunaga K. (1976), *A graph-theoretic approach to non-parametric Cluster Analysis*, I.E.E.E. Transactions on computers, vol. C-25, N. 9, september 1976.

[13] Pollet I. (1995), *Estimation d'un convexe à partir de données intérieures et extérieures dans le cas d'un processus de Poisson non homogène*, Mémoire de Licence, F.U.N.D.P. Namur, Belgique, inédit.

[14] Rasson J.P., Granville V. (1996), *Geometrical tools in classification*, inédit.

[15] Ripley B.D., Rasson J.-P. (1977), *Finding the edge of a Poisson forest*, Journal of Applied Probability, 14, pp. 483-499.

[16] Silverman B.W. (1986), *Density estimation for statistics and data analysis*, Chapman and Hall, London.

Annexes

Annexe A

Classification par détection de modes

A.1 Principe

Nous considérons n observations x_1, ..., x_n résultant d'un pocessus de Poisson non homogène sur la réunion de m domaines convexes disjoints.

Nous cherchons à détecter les modes de l'intensité du processus, qui a au préalable été estimée. Une fois ces modes détectés, nous tronquons pour chaque mode l'intensité de sorte qu'il ne reste qu'un nombre restreint d'observations autour du mode (qui n'est pas forcément une observation).
Nous pouvons alors considérer comme *noyau* de chaque classe :

1. Soit l'enveloppe convexe des observations contenues dans la partie tronquée,
2. Soit l'enveloppe convexe de la projection de la partie tronquée.

L'avantage de la deuxième solution réside dans le fait que, même lorsque le nombre d'observations est faible, elle reste valable, alors que la première solution n'a plus beaucoup de sens.

Il reste ensuite à affecter les observations n'appartenant pas aux noyaux à ceux-ci, en minimisant chaque fois l'intensité intégrée ajoutée au noyau.

Si le nombre de modes (i.e. le nombre de classes) obtenu est supérieur au nombre m

de classes voulu, il suffit ensuite de regrouper les classes en minimisant l'intensité intégrée sur les enveloppes.

A.2 Algorithme

a. Estimation de l'intensité.

b. Détection des modes.

c. Pour chaque mode,

1. Déterminer le seuil de troncature de sorte qu'il y ait "suffisamment" d'observations dans le noyau.
2. Calculer l'enveloppe convexe des points du noyau.

d. Pour tous les points non affectés,

1. Affecter celui qui minimise l'intensité intégrée ajoutée à un noyau.
2. Recalculer l'enveloppe convexe correspondante.

e. Regrouper des classes si nécessaire.

A.3 Détection des modes

L'idée utilisée est une généralisation donnée par Silverman [16] de Koontz, Narendra et Fukunaga [12].

Notons $\forall x_i, x_j,\ d_{ij} = \|x_i - x_j\|_\infty$.
Soit h le paramètre de lissage utilisé pour l'estimation $\hat{q}$ de l'intensité[1].

A partir d'une observation x_{i_1}, nous déterminons l'observation x_{i_2} qui maximise

$$\frac{\hat{q}(x_{i_2}) - \hat{q}(x_{i_1})}{d_{i_1 i_2}},$$

pour $d_{i_1 i_2} \leq h$ et $\hat{q}(x_{i_2}) > \hat{q}(x_{i_1})$, si elle existe.
Nous poursuivons en cherchant x_{i_3} qui maximise

$$\frac{\hat{q}(x_{i_3}) - \hat{q}(x_{i_2})}{d_{i_2 i_3}},$$

[1]Si jamais le processus donnait un nombre de classes inférieur à celui désiré, il suffirait de recommencer avec un h' plus petit.

si elle existe, etc.
En procédant ainsi, nous allons arriver à une observation x_{i_k} pour laquelle nous ne pourrons plus trouver de $x_{i_{k+1}}$. Nous considérerons alors que x_{i_k} est le "mode" d'une classe. C'est en fait une approximation de celui-ci, si le nombre de points est suffisamment grand.

Nous recommençons ensuite le processus avec une observation non encore utilisée. Finalement, nous obtenons les "modes" $y_1, ..., y_l$.

Annexe B

Code de CLUSTACC

```
C*****************************************************************
C*       Programme de clustering a 2 dimensions spatiales        *
C*                                                               *
C*                                          Nicolas JEANNEE      *
C*****************************************************************
      program clustacc

      implicit none
C ---------------------------------------------------------------------
C PARAMETRES:
C     nmaxpts: nombre de points maximum que peut contenir le fichier
C              de donnees.
C     taillemaxfen: taille maximale de la fenetre.
C VARIABLES:
C     nbrclvoulu: nombre de classes que l'utilisateur desire pour la
C                 classification finale.
C     nbclcr: nombre de classes creees par le programme au fur et a
C             mesure
C     nptsregroup: nombre de points regroupes par le programme.
C     indice: necessaire a la numerotation des classes creees.
C     h: vecteur contenant les rayons des boules centrees aux
C        observations.
C     x: les deux premieres lignes contiennent les coordonnees de
C        tous les points d'observation. La troisieme ligne est
C        creee par le programme et contient les numeros de classe
C        de chaque point d'observation.
C     densite: contient l'intensite estimee pour chaque point de la
C              fenetre.
C
C Remarque:-Les seules variables non reprises dans les descriptions
C           sont les indices de boucle.
C          -Les descriptions de variables des sous-routines ne
C           contiendront que les variables locales.
C ---------------------------------------------------------------------
      integer nbrclvoulu,nbclcr,nptsregroup,indice
      integer npts,nmaxpts
      integer taillefen,taillemaxfen
      parameter(nmaxpts=250,taillemaxfen=512)
```

```
      integer i,j
      real h(nmaxpts)
      integer densite(taillemaxfen,taillemaxfen)
      integer x(3,nmaxpts)
      logical debut

      common/pts/x
      common/dens/densite
      common/h/h

C -----------------------------------------------------------------
C -------------------Programme principal---------------------------
C -----------------------------------------------------------------
      call titre
      call prelim(npts,taillefen,nbrclvoulu)
      call estime_dens(npts,taillefen)

      nptsregroup=0
      nbclcr=0
      indice=0
      debut=.true.

 10   call increase(debut,npts,taillefen)

      do i=1,npts
         do j=i+1,npts
            if (nbclcr+npts-nptsregroup.le.nbrclvoulu) goto 20
            call regroup(i,j,npts,nptsregroup,nbclcr,indice)
         enddo
      enddo

      goto 10
 20   continue

C -----------------------------------------------------------------
C Il faut numeroter les points non regroupes, qui forment des
C classes a eux seuls.
C -----------------------------------------------------------------
      do i=1,npts
         if (x(3,i).eq.0) then
            indice=indice+1
            x(3,i)=indice
         endif
      enddo
      call resultat(npts,taillefen)
      end
C -----------------------------------------------------------------
C -------------------Sous-routine Titre----------------------------
C -----------------------------------------------------------------
      subroutine titre

      write(*,*)'   ********************************************'
      write(*,*)'   *  Programme de clustering a 2 dim. spatiales. *'
      write(*,*)'   *                              Nicolas Jeannee   *'
      write(*,*)'   ********************************************'
      write(*,*)' '
      write(*,*)' '
      end
```

```
C ---------------------------------------------------------------
C -----------------Sous-routine preliminaire---------------------
C ---------------------------------------------------------------
      subroutine prelim(npts,taillefen,nbrclvoulu)

C ---------------------------------------------------------------
C Cette sous-routine demande a l'utilisateur de rentrer le fichier
C de donnees a traiter, calcule le nombre de points npts de ce
C fichier, sa taille taillefen puis initialise le tableau de
C coordonnees x.
C Finalement, l'utilisateur doit rentrer le nombre de classes
C attendu nbrclvoulu.
C ---------------------------------------------------------------
      implicit none
C ---------------------------------------------------------------
C VARIABLE:
C     nomfich1: contient le nom du fichier de donnees.
C ---------------------------------------------------------------
      integer npts,nmaxpts,nbrclvoulu
      integer taillefen,taillemaxfen
      parameter(nmaxpts=250,taillemaxfen=512)
      integer i
      integer x(3,nmaxpts)
      character*20 nomfich1

      common/pts/x

 30   write(*,*) 'Entrez le nom du fichier de donnees(entre cotes):'
      read(*,*) nomfich1
      open(15,file=nomfich1,status='old')
      taillefen=0
      i=1
 40   read(15,*,end=50) x(1,i),x(2,i)
      x(1,i)=x(1,i)+40
      x(2,i)=x(2,i)+40
      write(*,*) x(1,i),' ',x(2,i)
      if (x(1,i).gt.taillefen) taillefen=x(1,i)
      if (x(2,i).gt.taillefen) taillefen=x(2,i)
      i=i+1
      goto 40
 50   close(15)
      npts=i-1
      taillefen=taillefen+40
      if ((taillefen.gt.taillemaxfen).or.(npts.gt.nmaxpts)) then
         write(*,*) 'Fichier trop grand et donc intraitable'
         goto 30
      endif
      write(*,*) 'Nombre de points du fichier:',npts
      write(*,*) 'Taille de la fenetre:',taillefen
      write(*,*) 'Le fichier a ete correctement lu...'

 60   write(*,*) 'Nombre de classes voulu:'
      read(*,*) nbrclvoulu
      if (nbrclvoulu.gt.npts) then
         write(*,*) 'Plus de classes que de points???'
         goto 60
      endif
      end
```

```
C ---------------------------------------------------------------
C -----------------Sous-routine Increase-------------------------
C ---------------------------------------------------------------
      subroutine increase(debut,npts,taillefen)

C ---------------------------------------------------------------
C Cette sous-routine se charge de l'augmentation du rayon des
C boules centrees aux points d'observation.
C ---------------------------------------------------------------
      implicit none

      integer npts,nmaxpts,taillefen,taillemaxfen
      parameter(nmaxpts=250,taillemaxfen=512)
      real h(nmaxpts)
      integer densite(taillemaxfen,taillemaxfen)
      integer d(nmaxpts)
      integer x(3,nmaxpts)
      integer inter(taillemaxfen,taillemaxfen)
      integer i,xx,yy
      logical debut

      common/dens/densite
      common/pts/x
      common/h/h
      common/inter/inter

      if (debut) then
         debut=.false.
         do i=1,npts
            h(i)=1.
         enddo
         do xx=1,taillefen
            do yy=1,taillefen
               inter(xx,yy)=0
            enddo
         enddo
      else
C ---------------------------------------------------------------
C On detecte d'abord les intersections entre les boules.
C ---------------------------------------------------------------
         do xx=1,taillefen
            do yy=1,taillefen
               if (inter(xx,yy).eq.1) inter(xx,yy)=0
            enddo
         enddo
         do i=1,npts
            do xx=x(1,i)-int(h(i)),x(1,i)+int(h(i))
               if ((xx.ge.1).and.(xx.le.taillefen)) then
                  do yy=x(2,i)-int(h(i)),x(2,i)+int(h(i))
                     if ((yy.ge.1).and.(yy.le.taillefen)) then
                        inter(xx,yy)=inter(xx,yy)+1
                     endif
                  enddo
               endif
            enddo
         enddo
C ---------------------------------------------------------------
C On calcule les intensites integrees sur les boules, en tenant
```

```
C compte des intersections.
C ------------------------------------------------------------------
         do i=1,npts
            d(i)=0.
            do xx=x(1,i)-int(h(i)),x(1,i)+int(h(i))
               if ((xx.ge.1).and.(xx.le.taillefen)) then
                  do yy=x(2,i)-int(h(i)),x(2,i)+int(h(i))
                     if ((yy.ge.1).and.(yy.le.taillefen)) then
                        if ((inter(xx,yy).eq.1.).and.
     +                       ((abs(xx-x(1,i)).eq.int(h(i))).or.
     +                       (abs(yy-x(2,i)).eq.int(h(i)))))) then
                           d(i)=d(i)+densite(xx,yy)
                        endif
                     endif
                  enddo
               endif
            enddo
C ------------------------------------------------------------------
C Augmentation du h.
C ------------------------------------------------------------------
      if (d(i).ne.0) h(i)=h(i)+1./d(i)
         enddo
      endif
      end
C ------------------------------------------------------------------
C -----Sous-routine d'estimation de la densite (Vincent B.)---------
C ------------------------------------------------------------------
      subroutine estime_dens(npts,taillefen)

C ------------------------------------------------------------------
C Cette sous-routine estime la densite des points situes dans la
C fenetre d'observation, en demandant a l'utilisateur s'il desire
C estimer le parametre de lissage rayon, ou s'il prefere le fixer.
C ------------------------------------------------------------------
      implicit none
C ------------------------------------------------------------------
C VARIABLES:
C     rayon: valeur du parametre de lissage, entre par l'utilisateur
C            ou estime par le programme.
C     rep: permet de lire le choix de l'utilisateur.
C ------------------------------------------------------------------
      integer rep
      integer npts,nmaxpts,taillefen,taillemaxfen
      parameter(nmaxpts=250,taillemaxfen=512)
      integer x(3,nmaxpts),rayon
      integer densite(taillemaxfen,taillemaxfen)

      common/pts/x
      common/dens/densite

      write(*,*) 'Debut d''estimation de la densite...'
      write(*,*) ' '
      write(*,*) 'Voulez-vous:'
      write(*,*) '  0:Fixer la valeur du h'
      write(*,*) '  1:L''estimer?'
  70  read(*,*) rep
      if (rep.eq.0) then
         write(*,*) 'Entrez la valeur de h:'
```

```
         read(*,*) rayon
         write(*,*) rayon
         call estimerdens(rayon,npts,taillefen)
      else
         if (rep.eq.1) then
            call estime_h(rayon,npts,taillefen)
            call estimerdens(rayon,npts,taillefen)
         else
            write(*,*) 'La reponse doit etre 0 ou 1!'
            goto 70
         endif
      endif
      write(*,*) 'Fin d''estimation de la densite'
      end
C ---------------------------------------------------------------
C ---Sous-routine d'estimation a proprement parler de la densite----
C ---------------------------------------------------------------
      subroutine estimerdens(rayon,npts,taillefen)

      implicit none

      integer xx,yy,i
      integer npts,nmaxpts,taillefen,taillemaxfen
      parameter(nmaxpts=250,taillemaxfen=512)
      integer x(3,nmaxpts),rayon
      integer densite(taillemaxfen,taillemaxfen)

      common/pts/x
      common/dens/densite

      do xx=1,taillefen
         do yy=1,taillefen
            densite(xx,yy)=0
         enddo
      enddo

      do i=1,npts
         do xx=x(1,i)-rayon,x(1,i)+rayon
            if ((xx.ge.1).and.(xx.le.taillefen)) then
               do yy=x(2,i)-rayon,x(2,i)+rayon
                  if ((yy.ge.1).and.(yy.le.taillefen))
     +                 densite(xx,yy)=densite(xx,yy)+1
               enddo
            endif
         enddo
      enddo
      end
C ---------------------------------------------------------------
C ---------Sous-routine d'estimation du parametre h-----------------
C ---------------------------------------------------------------
      subroutine estime_h(rayon,npts,taillefen)

      implicit none
C ---------------------------------------------------------------
C VARIABLES:
C     sum: permet de sommer des intensites.
C     kl: contient la valeur du critere.
C     klmax: contient la valeur maximale du critere.
```

```
C     h: indice de boucle sur toutes les valeurs du parametre de
C        lissage.
C -------------------------------------------------------------------
      integer npts,nmaxpts,taillefen,taillemaxfen
      parameter (nmaxpts=250,taillemaxfen=512)
      real sum,klmax,kl
      integer x(3,nmaxpts)
      integer rayon,h
      integer i,j,xx,yy
      integer densite(taillemaxfen,taillemaxfen)

      common/pts/x
      common/dens/densite

      write(*,*) 'Estimation du h...'

      rayon=0
      klmax=-1.E25
      do h=1,60
         call estimerdens(h,npts,taillefen)

         kl=0      ! kl est deja considere comme un log...
         sum=0
         do xx=1,taillefen
            do yy=1,taillefen
               sum=sum+densite(xx,yy)
            enddo
         enddo

         do i=1,npts
            do xx=x(1,i)-h,x(1,i)+h
               do yy=x(2,i)-h,x(2,i)+h
                  if ((xx.ge.1).and.(xx.le.taillefen).and.
     +                (yy.ge.1).and.(yy.le.taillefen))
     +                densite(xx,yy)=densite(xx,yy)-1
               enddo
            enddo
            sum=0
            do xx=1,taillefen
               do yy=1,taillefen
                  sum=sum+densite(xx,yy)
               enddo
            enddo
            if (densite(x(1,i),x(2,i)).ne.0) then
               kl=kl+log(1.*densite(x(1,i),x(2,i)))-log(sum)
            else
               kl=-1.E25
            endif
            do xx=x(1,i)-h,x(1,i)+h
               do yy=x(2,i)-h,x(2,i)+h
                 if ((xx.ge.1).and.(xx.le.taillefen).and.
     +                (yy.ge.1).and.(yy.le.taillefen))
     +                densite(xx,yy)=densite(xx,yy)+1
               enddo
            enddo
         enddo

         if (klmax.le.kl) then
```

```
            rayon=h
            klmax=kl
            write(*,*) 'Nouvel h :',rayon
            write(*,*) 'Valeur de KL:',kl
         else
            write(*,*) 'Produit :',kl
         endif

      enddo
      write(*,*) 'Estimation: ',rayon
      end
C -----------------------------------------------------------------
C -----------------Sous-routine de regroupement--------------------
C -----------------------------------------------------------------
      subroutine regroup(i,j,npts,nptsregroup,nbclcr,indice)

      implicit none
C -----------------------------------------------------------------
C VARIABLES :
C     cl1,cl2: permettent le changement de numero d'une classe.
C -----------------------------------------------------------------
      integer nbclcr,nptsregroup,indice
      integer npts,nmaxpts
      parameter(nmaxpts=250)
      integer i,j,k,cl1,cl2
      real h(nmaxpts)
      integer x(3,nmaxpts)

      common/pts/x
      common/h/h

      if ((i.ne.j).and.((x(3,i).ne.x(3,j)).or.
     +     (x(3,i).eq.0))) then
C -----------------------------------------------------------------
C Les 2 points n'appartiennent pas a la meme classe, ou un des 2
C points au moins n'est pas encore affecte a une classe. On ne
C compare pas non plus un point a lui-meme(!).
C -----------------------------------------------------------------
         if ((abs(x(1,i)-x(1,j)).le.int(h(i)+h(j))).and.
     +        (abs(x(2,i)-x(2,j)).le.int(h(i)+h(j)))) then
C -----------------------------------------------------------------
C Les points i et j doivent etre regroupes.
C -----------------------------------------------------------------
            if (x(3,i).eq.0) then
               if (x(3,j).eq.0) then
C -----------------------------------------------------------------
C Vu qu'aucun des 2 points n'est encore affecte a une classe, on en
C cree une nouvelle, qui contient ces 2 points.
C -----------------------------------------------------------------
                  nbclcr=nbclcr+1
                  indice=indice+1
                  x(3,i)=indice
                  x(3,j)=indice
                  nptsregroup=nptsregroup+2
               else
C -----------------------------------------------------------------
C Le point j appartenant deja a une classe, on doit juste
C affecter i a la classe de j.
```

```
C ---------------------------------------------------------------
                  nptsregroup=nptsregroup+1
                  x(3,i)=x(3,j)
               endif
            else
C ---------------------------------------------------------------
C Le point i appartenant deja a une classe, on doit juste
C affecter j a la classe de i.
C ---------------------------------------------------------------
               if (x(3,j).eq.0) then
                  nptsregroup=nptsregroup+1
                  x(3,j)=x(3,i)
               else
C ---------------------------------------------------------------
C Les 2 points appartiennent a des classes differentes, qu'il faut
C donc regrouper. Pour cela, on determine laquelle des 2 classes a
C le plus petit indice, et on affecte de cet indice de classe tous
C les points de l'autre classe.
C ---------------------------------------------------------------
                  nbclcr=nbclcr-1
                  if (x(3,j).gt.x(3,i)) then
                     cl1=x(3,j)
                     cl2=x(3,i)
                  else
                     cl1=x(3,i)
                     cl2=x(3,j)
                  endif
                  do k=1,npts
                     if(x(3,k).eq.cl1) then
                        x(3,k)=cl2
                     endif
                  enddo
               endif
            endif
         endif
      endif
      end
C ---------------------------------------------------------------
C ------------------Sous-routine resultat------------------------
C ---------------------------------------------------------------
      subroutine resultat(npts,taillefen)

C ---------------------------------------------------------------
C Sous-routine qui initialise et complete les matrices contenant la
C classification finale.
C ---------------------------------------------------------------
      implicit none
C ---------------------------------------------------------------
C VARIABLES:
C     nomfich2: contient le nom du fichier de sortie.
C     result: matrice contenant les numeros de classe de tous les
C             points d'observation.
C ---------------------------------------------------------------
      integer npts,nmaxpts,taillefen,taillemaxfen
      parameter(nmaxpts=250,taillemaxfen=512)
      integer x(3,nmaxpts)
      integer i,j
      character*20 nomfich2
```

```
      byte result(taillemaxfen,taillemaxfen)

      common/pts/x

      do i=1,taillefen
         do j=1,taillefen
            result(i,j)=0
         enddo
      enddo
      write(*,*) 'Classification finale:'
      do i=1,npts
         write(*,*) x(1,i),'  ',x(2,i),'  ',x(3,i)
         result(x(1,i),x(2,i))=x(3,i)
      enddo
c mettre result dans un fichier output!!!
      write(*,*) 'Entrez le nom du fichier resultat(entre cotes):'
      read(*,*) nomfich2
      open(25,file=nomfich2,status='unknown',form='formatted',recordtype
     +     ='fixed',recl=taillemaxfen)
      do i=1,taillemaxfen
         write(25,rec=i) (result(i,j),j=1,taillemaxfen)
      enddo
      close(25)

      write(*,*) 'Vous pouvez visionner votre resultat dans le'
      write(*,*) 'fichier resultat en executant: '
      write(*,*) 'visu1 -i resultat -o fichier visionnable'
      write(*,*) 'puis xv.'
      end
C ---------------------------------------------------------------------
C ---------------------------------------------------------------------
```

Annexe C

Code d'ECHANGE

Les sous-routines LECFICH, ESTIME_DENS et RESULTAT sont identiques à celles de CLUSTACC. Elles ne sont donc pas reprises ici.
La sous-routine CONVEXE a été implémentée par Laurent Waerenburgh, dans le cadre d'un mémoire en Statistique aux FUNDP, en 1990.

```
      program echange

      implicit none

      integer npts,nmaxpts,nbcla,nbclamax
      integer taillefen,taillemaxfen
      parameter(nmaxpts=100,taillemaxfen=512,nbclamax=10)
      integer i,j,k,pt
      integer x(3,nmaxpts)
      integer intint(nbclamax),inttot,intnouv,intmin
      integer chang(3,0:nmaxpts)
      integer chtri(0:nmaxpts)
      integer densite(taillemaxfen,taillemaxfen)
      integer class,nptcl,clpt
      logical ok

      external convexe
      integer nhmax,nfmax,npmx1,npmax
      parameter(nhmax=500,nfmax=600,npmx1=500,npmax=2000)
      integer point(2,nmaxpts)
      integer face(2,nfmax)
      integer point1(nmaxpts),hyper1(3,nhmax),face1(nfmax)
      integer dim,nh,nf
      double precision hyperplan(2,nhmax),vectra(2),volume
      include 'fatal_limits.def'

      integer l,rep
      double precision hyp1(nbclamax,2,nhmax)
      integer hyp2(nbclamax,nhmax)
      real xbarre(nbclamax),ybarre(nbclamax)
      real xsav,ysav
```

```
      common/pts/x
      common/dens/densite
      common/conv/hyperplan,hyper1,nh,point1,point
      common/hyp/hyp1,hyp2
      common/barre/xbarre,ybarre
C-----------------------------------------------------------------------
C---------------------Programme principal-------------------------------
C-----------------------------------------------------------------------
      call lecfich(npts,taillefen,nbcla)
      call estime_dens(npts,taillefen)

c ----------------
c initialisations.
c ----------------
 10   do i=1,3
         do j=0,npts
            chang(i,j)=0
         enddo
      enddo
      do i=1,nbcla
         do j=1,nhmax
            do k=1,2
               hyp1(i,k,j)=0
            enddo
            hyp2(i,j)=0
         enddo
      enddo

c ---------------------------------------------------------------
c calcul des enveloppes convexes, et des int. int. sur celles-ci.
c ---------------------------------------------------------------
      do i=1,nbcla
         call calc_conv(npts,nptcl,i,0)
         call sommets(nptcl,i,chang)
         call calc_int(taillefen,nptcl,i,intint(i))
      enddo

      inttot=0
      do l=1,nbcla
         inttot=inttot+intint(l)
      enddo
      write(*,*) 'Intensite totale:',inttot

c -----------------------------------------------------------------------
c les sommets d'enveloppe sont ordonnes par int. int. ajoutee corissante.
c on considere a chaque etape la reaffectation du sommet pour lequel
c cette int. ajoutee est maximale.
c -----------------------------------------------------------------------
      call ordonne(npts,taillefen,nbcla,chang,chtri,intint)

 15   if (chtri(0).eq.0) goto 20

      do j=1,npts
         if ((x(1,j).eq.chang(1,chtri(chtri(0)))).and.(x(2,j)
     +        .eq.chang(2,chtri(chtri(0)))) pt=j
      enddo

      chtri(0)=chtri(0)-1
```

```
      clpt=x(3,pt)

      call calc_conv(npts,nptcl,clpt,pt)
      call calc_int(taillefen,nptcl,clpt,intnouv)

      class=clpt
      intmin=intint(clpt)-intnouv
      intint(clpt)=intnouv

c ---------------------------------------------------------------------
c on determine la classe pour laquelle le point pt considere minimise
c l'int. int. ajoutee.
c ---------------------------------------------------------------------
      do j=1,nbcla
         if (j.ne.clpt) then
            x(3,pt)=j
            call calc_conv(npts,nptcl,j,0)
            xsav=xbarre(j)
            ysav=ybarre(j)
            call calc_int(taillefen,nptcl,j,intnouv)
            call accept(taillefen,nbcla,ok,j)

            if (ok.eq..true.) then
               if (intnouv-intint(j).lt.intmin) then
                  class=j
                  intmin=intnouv-intint(j)
               endif
            endif

c ---------------------------------------------------------------------
c apres avoir essaye d'affecter pt a une classe, il faut recalculer
c l'enveloppe de cette classe sans pt, ainsi que le point int.:
c ---------------------------------------------------------------------
            call calc_conv(npts,nptcl,j,pt)
            xbarre(j)=xsav
            ybarre(j)=ysav
         endif
      enddo

c ----------------------------------------
c on affecte le point a la classe appropriee.
c ----------------------------------------
      x(3,pt)=class
      intint(x(3,pt))=intint(x(3,pt))+intmin

      if (class.ne.clpt) then
         write(*,*) 'Changement de classe du point',x(1,pt),x(2,pt)
         write(*,*) 'de la classe',clpt,' vers',class
         goto 10
      else
         goto 15
      endif

 20   write(*,*) 'Int. integree finale:',inttot

      call resultat(npts,taillefen)
      end
C---------------------------------------------------------------------
```

```
C-------Sous-routine de calcul d'enveloppe convexe d'une classe-----------
C---------------------------------------------------------------------
      subroutine calc_conv(npts,nptcl,cl,pt)

      integer nptcl,cl,pt,i,j
      integer npts,nmaxpts,nbclamax
      parameter (nmaxpts=100,nbclamax=10)
      integer x(3,nmaxpts)

      integer nhmax,nfmax,npmx1,npmax
      parameter(nhmax=500,nfmax=600,npmx1=500,npmax=2000)
      integer point(2,nmaxpts)
      integer face(2,nfmax)
      integer point1(nmaxpts),hyper1(3,nhmax),face1(nfmax)
      integer dim,nh,nf
      double precision hyperplan(2,nhmax),vectra(2),volume
      include 'fatal_limits.def'

      double precision hyp1(nbclamax,2,nhmax)
      integer hyp2(nbclamax,nhmax)

      common/pts/x
      common/conv/hyperplan,hyper1,nh,point1,point
      common/hyp/hyp1,hyp2

      fatal_limit_nhmax=nhmax-1
      fatal_limit_nfmax=nfmax-2
      fatal_limit_npmax=npmax-npmx1
      fatal_current_nh=0
      fatal_current_nf=0

      nptcl=0     !nptcl contiendra le nombre de points de la classe cl
      do i=1,2
         do j=1,npts
            point(i,j)=0
         enddo
      enddo
      do i=1,npts
         if((x(3,i).eq.cl).and.(i.ne.pt)) then
            nptcl=nptcl+1
            point(1,nptcl)=x(1,i)
            point(2,nptcl)=x(2,i)
         endif
      enddo

      call convexe(point,hyperplan,vectra,face,point1,hyper1,face1,2,
     +     nptcl,nh,nf,volume,2,'false')

      do i=1,2
         do j=1,nhmax
            hyp1(cl,i,j)=hyperplan(i,j)
            hyp2(cl,j)=hyper1(1,j)
         enddo
      enddo
      end
C---------------------------------------------------------------------
C-------------Sous-routine de traitement des sommets------------------
C---------------------------------------------------------------------
```

```
      subroutine sommets(nptcl,cl,tab)

      integer nptcl,cl,i,j
      integer nmaxpts
      parameter (nmaxpts=100)
      integer tab(3,0:nmaxpts)

      integer nhmax,nfmax,npmx1,npmax
      parameter(nhmax=500,nfmax=600,npmx1=500,npmax=2000)
      integer point(2,nmaxpts)
      integer face(2,nfmax)
      integer point1(nmaxpts),hyper1(3,nhmax),face1(nfmax)
      integer dim,nh,nf
      double precision hyperplan(2,nhmax),vectra(2),volume
      include 'fatal_limits.def'

      common/conv/hyperplan,hyper1,nh,point1,point

      i=tab(1,0)
      do j=1,nptcl
         if (point1(j).eq.1) then
            i=i+1
            tab(1,i)=point(1,j)
            tab(2,i)=point(2,j)
            tab(3,i)=cl
         endif
      enddo
      tab(1,0)=i
      end
C-------------------------------------------------------------------------
C-------Sous-routine de calcul d'intensite integree sur une classe--------
C-------------------------------------------------------------------------
      subroutine calc_int(taillefen,nptcl,cl,res)

      implicit none

      integer cl,res,nptcl
      real compt
      integer i,xx,yy,taillefen
      integer taillemaxfen,nmaxpts
      integer nbclamax
      parameter (nmaxpts=100,taillemaxfen=512,nbclamax=10)
      real xbarre(nbclamax),ybarre(nbclamax)
      integer densite(taillemaxfen,taillemaxfen)
      logical app_conv

      integer nhmax,nfmax,npmx1,npmax
      parameter(nhmax=500,nfmax=600,npmx1=500,npmax=2000)
      integer point(2,nmaxpts)
      integer face(2,nfmax)
      integer point1(nmaxpts),hyper1(3,nhmax),face1(nfmax)
      integer dim,nh,nf
      double precision hyperplan(2,nhmax),vectra(2),volume
      include 'fatal_limits.def'

      common/conv/hyperplan,hyper1,nh,point1,point
      common/dens/densite
      common/barre/xbarre,ybarre
```

```
      res=0
      compt=0.
      xbarre(cl)=0
      ybarre(cl)=0

      do i=1,nptcl
         if (point1(i).eq.1) then
            compt=compt+1
            xbarre(cl)=xbarre(cl)+point(1,i)
            ybarre(cl)=ybarre(cl)+point(2,i)
         endif
      enddo

      xbarre(cl)=xbarre(cl)/compt
      ybarre(cl)=ybarre(cl)/compt

      do xx=1,taillefen
         do yy=1,taillefen
            if (app_conv(xx,yy,cl).eq..true.) res=res+densite(xx,yy)
         enddo
      enddo
      end
C-------------------------------------------------------------------------
C-----------Fonction d'appartenance a une enveloppe convexe---------------
C-------------------------------------------------------------------------
      logical function app_conv(x,y,cl)

      implicit none

      integer i,x,y
      integer cl
      real xl,yl
      integer signe,signpt,signptint
      integer nmaxpts,nbclamax
      parameter (nmaxpts=100,nbclamax=10)
      real xbarre(nbclamax),ybarre(nbclamax)

      integer nhmax,nfmax,npmx1,npmax
      parameter(nhmax=500,nfmax=600,npmx1=500,npmax=2000)
      integer point(2,nmaxpts)
      integer face(2,nfmax)
      integer point1(nmaxpts),hyper1(3,nhmax),face1(nfmax)
      integer dim,nh,nf
      double precision hyperplan(2,nhmax),vectra(2),volume
      include 'fatal_limits.def'

      common/conv/hyperplan,hyper1,nh,point1,point
      common/barre/xbarre,ybarre

      xl=x
      yl=y
      app_conv=.true.

      do i=1,nh
         signpt=signe(xl,yl,i,cl)
         signptint=signe(xbarre(cl),ybarre(cl),i,cl)
         if ((signpt.ne.0).and.(signptint.ne.signpt))
```

```
     +        app_conv=.false.
      enddo
      end
C-------------------------------------------------------------------------
C-----------------Fonction de calcul de signe-----------------------------
C-------------------------------------------------------------------------
      integer function signe(xx,yy,num,cl)

      implicit none

      integer num,cl
      real xx,yy,valeur
      integer nmaxpts
      integer nbclamax
      integer nhmax
      parameter (nmaxpts=100,nbclamax=10)
      parameter(nhmax=500)
      double precision hyp1(nbclamax,2,nhmax)
      integer hyp2(nbclamax,nhmax)

      common/hyp/hyp1,hyp2

      valeur=hyp1(cl,1,num)*xx+hyp1(cl,2,num)*yy-hyp2(cl,num)
      if (valeur.gt.0.0001) then
         signe=1
      else if (valeur.lt.-0.0001) then
         signe=-1
      else
         signe=0
      endif
      end
C-------------------------------------------------------------------------
C------Sous-routine d'acceptation d'affectation a une classe--------------
C-------------------------------------------------------------------------
      subroutine accept(taillefen,nbcla,ok,num)
C -----------------------------------------------------------------------
C But : Voir que l'enveloppe de la classe cl construite n'en
C       intersecte pas une autre.
C -----------------------------------------------------------------------
      implicit none

      integer taillefen,nbcla
      logical ok
      integer num
      logical app_conv
      integer xx,yy,i

      ok=.true.
      do xx=1,taillefen
         do yy=1,taillefen
            if (app_conv(xx,yy,num).eq..true.) then
               do i=1,nbcla
                  if ((ok.eq..true.).and.(i.ne.num)) then
                     if (app_conv(xx,yy,i).eq..true.) then
                        ok=.false.
                     endif
                  endif
               enddo
```

```
            endif
          enddo
        enddo
        end
C-------------------------------------------------------------------------
C------------Sous-routine de choix du point a considerer------------------
C-------------------------------------------------------------------------
        subroutine ordonne(npts,taillefen,nbcl,tab,tabtri,intint)

        implicit none

        integer i,j,k,pt,min,nbcl,nbclamax,taillefen
        integer npts, nmaxpts,nptcl,clpt
        parameter (nmaxpts=100,nbclamax=10)
        integer tab(3,0:nmaxpts)
        integer tabtri(0:nmaxpts)
        integer intens(nmaxpts),intnouv
        integer intint(nbclamax)
        integer x(3,nmaxpts)

        integer nhmax,nfmax,npmx1,npmax
        parameter(nhmax=500,nfmax=600,npmx1=500,npmax=2000)
        integer point(2,nmaxpts)
        integer face(2,nfmax)
        integer point1(nmaxpts),hyper1(3,nhmax),face1(nfmax)
        integer dim,nh,nf
        double precision hyperplan(2,nhmax),vectra(2),volume
        include 'fatal_limits.def'

        common/pts/x
        common/conv/hyperplan,hyper1,nh,point1,point

c -------------------------------------------------------------
c on determine, pour chacun des sommets des enveloppes convexes,
c l'intensite integree qu'il ajoute a sa propre classe.
c -------------------------------------------------------------
        do j=1,tab(1,0)

           do k=1,npts
              if ((tab(1,j).eq.x(1,k)).and.(tab(2,j).eq.x(2,k))) then
                 pt=k
              endif
           enddo

           call calc_conv(npts,nptcl,tab(3,j),pt)
           call calc_int(taillefen,nptcl,tab(3,j),intnouv)
           intens(j)=intint(tab(3,j))-intnouv

        enddo

c -----------------------------------------------------
c on trie les sommets par intensite ajoutee croissante.
c -----------------------------------------------------
        tabtri(0)=tab(1,0)
        do i=1,tab(1,0)
           tabtri(i)=0
           min=100000
           do j=1,tab(1,0)
```

```
            if (intens(j).lt.min) then
               min=intens(j)
               tabtri(i)=j
            endif
         enddo
         intens(tabtri(i))=100000
      enddo
      end
C-----------------------------------------------------------------------
C-----------------------------------------------------------------------
```

Annexe D

Code de ECHGEOM

Les sous-routines relatives au traitement des enveloppes convexes sont identiques à celles utilisées par ECHANGE.

```
      program echgeom

      implicit none

      integer npts,nmaxpts,nbcla,nbclamax
      integer taillefen,taillemaxfen
      parameter(nmaxpts=100,taillemaxfen=512,nbclamax=10)
      integer i,j,k,pt,cl
      integer x(3,nmaxpts)
      real z(0:nmaxpts)
      integer y(nmaxpts)
c             y contient une copie des numeros de classe des points, qui
c             est mise a jour apres chaque enveloppe.
c             z contient egalement une copie des num. de classe des
c             points, mais qui contient a chaque iteration les num. de
c             classe correspondant a la classif. la plus optimale trouvee.
c             z(0) contient l'int.int. de la classif la plus opt. trouvee.
      real intint(nbclamax),inttot,intnouv,intens
      integer chang(3,0:nmaxpts)
      real densite(taillemaxfen,taillemaxfen)
      integer nptcl
      real rayon
      integer rep,xx,yy
      logical nochang,first,vide(nbclamax)
      real ax,ay,ker

      external convexe

      integer nhmax,nfmax,npmx1,npmax
      parameter(nhmax=500,nfmax=600,npmx1=500,npmax=2000)
      integer point(2,nmaxpts)
      integer face(2,nfmax)
      integer point1(nmaxpts),hyper1(3,nhmax),face1(nfmax)
      integer dim,nh,nf
      double precision hyperplan(2,nhmax),vectra(2),volume
```

```
            write(*,*) 'change de classe.'
            intens=inttot
            first=.false.
            do i=1,npts
               z(i)=x(3,i)
            enddo
            z(0)=inttot
            goto 5
         endif
      else
         write(*,*) 'On affecte le pt corresp.:',x(1,pt),x(2,pt)
         write(*,*) 'a la classe',cl
         write(*,*) '  '

         x(3,pt)=cl
         if (cl.ne.y(pt)) nochang=.false.

         call calc_conv(npts,nptcl,cl)

         do xx=1,taillefen
            do yy=1,taillefen
               ax=i-rayon
               ay=i-rayon
               if (xx.ne.x(1,pt))
     +              ax=.5*(1-rayon)**2*rayon**abs(xx-x(1,pt))
               if (yy.ne.x(2,pt))
     +              ay=.5*(1-rayon)**2*rayon**abs(yy-x(2,pt))
               ker=ax*ay
               densite(xx,yy)=densite(xx,yy)+ker
            enddo
         enddo

         inttot=0
         do i=1,nbcla
            call calc_conv(npts,nptcl,i)
            call calc_int(taillefen,nptcl,i,intint(i))
            inttot=inttot+intint(i)
         enddo
         goto 10
      endif

      do i=1,npts
         x(3,i)=z(i)
      enddo
      inttot=z(0)

      write(*,*) ' '
      write(*,*) 'Le processus s''est stabilise'
      write(*,*) 'Int. integree finale:',inttot
      call resultat(npts,taillefen)
      end
C-----------------------------------------------------------------------
C--------------------Sous-routine de titre------------------------------
C-----------------------------------------------------------------------
      subroutine titre

      write(*,*) '  '
      write(*,*) ' ********************************************** '
```

```
      write(*,*) ' * Programme d''optimisation d''une classif.     * '
      write(*,*) ' *  initiale par reallocation successive des   * '
      write(*,*) ' *      sommets des enveloppes convexes        * '
      write(*,*) ' *                                  M.Jeannee  * '
      write(*,*) ' ********************************************** '
      write(*,*) '  '
      write(*,*) '  '
      end
C ---------------------------------------------------------------
C ----------Sous-routine d'estimation de la densite--------------
C ---------------------------------------------------------------
      subroutine estimerdens(h,npts,taillefen,i)

      implicit none

      integer xx,yy,i,j
      integer npts,nmaxpts,taillefen,taillemaxfen
      parameter(nmaxpts=250,taillemaxfen=512)
      integer x(3,nmaxpts)
      real h
      real densite(taillemaxfen,taillemaxfen)
      real ax,ay,ker

      common/pts/x
      common/dens/densite

      do xx=1,taillefen
         do yy=1,taillefen
            densite(xx,yy)=0
         enddo
      enddo
      do j=1,npts
        if (j.ne.i) then
            do xx=1,taillefen
               do yy=1,taillefen
                  ax=1-h
                  ay=1-h
                  if (xx.ne.x(1,j))
     +                 ax=.5*(1-h)**2*h**abs(xx-x(1,j))
                  if (yy.ne.x(2,j))
     +                 ay=.5*(1-h)**2*h**abs(yy-x(2,j))
                  ker=ax*ay
                  densite(xx,yy)=densite(xx,yy)+ker
               enddo
            enddo
         endif
      enddo
      end
C---------------------------------------------------------------------
C------------Sous-routine de choix du point a affecter----------------
C---------------------------------------------------------------------
      subroutine minim(npts,taillefen,nbcla,inttot,intint,pt,cl)

      implicit none

      integer i,j,k,pt,cl,nbcla,nbclamax,taillefen,taillemaxfen
      parameter(taillemaxfen=512)
      integer xx,yy
```

```
      integer npts,nmaxpts,nptcl,clpt
      parameter (nmaxpts=100,nbclamax=10)
      integer x(3,nmaxpts)
      real inttot,intnouv
      real intint(nbclamax)
      real intmin,intcoumin,clcoumin
      real xbarre(nbclamax),ybarre(nbclamax)
      real xsav,ysav
      logical ok

      common/pts/x
      common/barre/xbarre,ybarre

      pt=0
      cl=0
      intmin=inttot

      do i=1,npts
         if (x(3,i).eq.-1) then
            clcoumin=0
            intcoumin=inttot
            do j=1,nbcla
               xsav=xbarre(j)
               ysav=ybarre(j)
               x(3,i)=j

               call calc_conv(npts,nptcl,j)
               call calc_int(taillefen,nptcl,j,intnouv)

               call accept(taillefen,nbcla,ok,j)

               if (ok.eq..true.) then
                  if (intnouv-intint(j).lt.intcoumin) then
                     clcoumin=j
                     intcoumin=intnouv-intint(j)
                  endif
               endif

               x(3,i)=-1

               call calc_conv(npts,nptcl,j)
               xbarre(j)=xsav
               ybarre(j)=ysav
            enddo
            if (intcoumin.lt.intmin) then
               intmin=intcoumin
               cl=clcoumin
               pt=i
            endif
         endif
      enddo
      if (pt.ne.0) write(*,*) 'Int. min. aj.:',intmin,'
     +     par le pt',pt
      end
C-----------------------------------------------------------------------
C-----------------------------------------------------------------------
```